DEPARTMENT OF THE ARMY
HEADQUARTERS UNITED STATES ARMY MATERIEL COMMAND
WASHINGTON, DC, 20310

DEPARTMENT OF THE NAVY
HEADQUARTERS NAVAL MATERIAL COMMAND
WASHINGTON, DC, 20360

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR FORCE LOGISTICS COMMAND
WRIGHT-PATTERSON AFB, OHIO 45433

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR FORCE SYSTEMS COMMAND
ANDREWS AFB, WASHINGTON, DC, 20331

JOINT TECHNICAL COORDINATING GROUP
FOR
AIRCRAFT SURVIVABILITY

18 July 1983

FROM: C. Padgett - JTCG/AS DCIIS Member
TO: Distribution

SUBJ: Modified QKLOOK Program

ENCL: (1) Change 1 to JTCG/AS-79-V-008
(2) SUBROUTINE ASPTONIND

REF: (a) JTCG/AS-79-V-008

1. Recently, the four (4) QKLOOK programs, in reference (a), have been extensively modified. The modifications were made to increase the usefulness and ease of use of the QKLOOK model. The changes made (1) increase the user's control of the $P_{K/H}$ functions used in QKLOOK, (2) allow the user to select true or incremental vulnerable areas, and (3) brought the programs in line with the FORTRAN 77 standard. All the changes are thoroughly documented in enclosure (1).

2. In addition, the program VAMERGE which re-formats the QKLOOK output into a form usable by the ASALT program has now been documented. This documentation is enclosed as enclosure (2).

3. If you have any questions, contact:

Andrew Kilikauskas
Tele: AV 437-3681
(619) 939-3681

Connie Padgett

Connie Padgett, JTCG/AS
Documentation Publication Engineer

83 07 28 048

```
*AD-B049 011L     19/4, 9/2, 20/5
 ARMAMENT SYSTEMS INC   ANAHEIM CA
                                              80-19
 QKLOOK COMPUTER PROGRAMS ANALYST AND USER
 MANUAL.
 FINAL REPT. OCT 78-DEC 79,
 STEENROD,FREDERICK J. ; MUSCH,JOHN F. ;
    JUN 80   422P  CONTRACT
 N00123-76-C-0960  JTCG/AS  79-V-008
                                   UNCLASSIFIED REPORT
 DISTRIBUTION LIMITED TO U.S. GOVT. AGENCIES
 ONLY; TEST AND EVALUATION; JUL 77. OTHER
 REQUESTS FOR THIS DOCUMENT MUST BE REFERRED TO
 COMMANDER, NAVAL AIR SYSTEMS COMMAND, ATTN:
 AIR-5184J, WASHINGTON, DC 20361.
```

Code 1"

JTCG/AS-79-V-008

DISTRIBUTION LIST

Aeronautical Systems Division (AFSC)
Wright-Patterson AFB, OH 45433
Attn: ASD/XROT (G. B. Bennett)

Air Force Wright Aeronautical Laboratories
Wright-Patterson AFB, OH 45433
Attn: AFWAL/FIES (CDIC) (2 copies)
Attn: AFWAL/FIES (J. Hodges)
Attn: AFWAL/FIESD (G. Streets)
Attn: AFWAL/FIMB (P. Waggoner)

Applied Technology Laboratory
Army Research & Technology Laboratory (AVRADCOM)
Ft. Eustis, VA 23604
Attn: DAVDL-ATL-AL (Mr. Merritt)

Army Materials and Mechanics Research Center
Watertown, MA 02172
Attn: DRXMR-RA (T. V. Hynes) (2 copies)

David W. Taylor Naval Ship R&D Center
Carderock Laboratory
Bethesda, MD 20084
Attn: Code 1740.2 (F. J. Fisch)

Defense Technical Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: DTIC-TCA (2 copies)

Deputy Chief of Staff (AIR)
Marine Corps Headquarters
Washington, DC 20380
Attn: LT COL Huckleberry (ATW-22)

ERADCOM
Fort Monmouth, NJ 07703
Attn: DELSD-EV (C. Goldy)

Naval Air Systems Command
Washington, DC 20361
Attn: AIR-5164J (2 copies)

Naval Postgraduate School
Monterey, CA 93940
Attn: Code 67BP (R. E. Ball)

See: AD-B049 011, Jtd. Jun '80 basic rpt.

Naval Weapons Center
China Lake, CA 93555
Attn: Code 3381 (C. Padgett) (2 copies)

Naval Weapons Support Center
Crane, IN 47522
Attn: Code 502 (N. L. Papke)

Applications Research Corp.
330 S. Ludlow St.
Dayton, OH 45402

Armanent Systems, Inc.
712-F North Valley Street
Anaheim, CA 92801
Attn: J. Musch

The BDM Corp.
1801 Randolph S.E.
Albuquerque, NM 87106
Attn: A. J. Holten

Bell Helicopter Textron
Division of Textron Inc.
P.O. Box 482
Fort Worth, TX 76101
Attn: Security/Dept. 12, J. R. Johnson

The Boeing Aerospace Company
P.O. Box 3999
Seattle, WA 98124
Attn: C. J. Artura, M/S 8C-23

The Boeing Vertol Company
Boeing Center
P.O. Box 16858
Philadelphia, PA 19142
Attn: N. Caravasos, M/S P32-18

Boeing Military Airplane Company
P.O. Box 3707
Seattle, WA 98124
Attn: K. F. Brettmann, M/S 41-10 (2 copies)

Boeing Military Airplane Co.
3801 S. Oliver St.
Wichita, KS 67210
Attn: M/S K16-14

Calspan Corp.
P.O. Box 235
Buffalo, NY 14221
Attn: Library (V. M. Young)

Denver Research Institute
University of Denver
University Park Station
2360 S. Gaylord Street
Denver, CO 80210
Attn: R. F. Recht

Falcon Research and Development Co.
2350 Alamo Ave., SE
Albuquerque, NM 87106
Attn: Dan Dunbar

General Dynamics Corp.
Fort Worth Division
Grants Lane, P.O. Box 748
Fort Worth, TX 76101
Attn: Dr. Clyde Boykin - Mail Zone 2654

Grumman Aerospace Corp.
South Oyster Bay Rd.
Bethpage, NY 11714
Attn: J. P. Archey Jr., Dept. 662, Mail 31-05 Attn: H.B.Smith TIC,Plant 35
Attn: J. Hartung, Dept. 661, C27-05

Hughes Helicopters
Centinela Ave. & Teal St.
Culver City, CA 90230
Attn: Library, 2/T2124 (D. K. Goss)

Lockheed-Georgia Co.
A Division of Lockheed Aircraft Corp.
86 S. Cobb Drive
Marietta, GA 30063
Attn: Sci-Tech Info Center, 72-34 Zone 26 (T. J. Kopkin)
Attn: Dr. Gary Jackman 72-08, Zone 415

McDonnell Douglas Corp.
P.O. Box 516
St. Louis, MO 63166
Attn: R. D. Detrich, Dept. 022
Attn: Michael Meyers, Dept. 345, Bldg. 33, Post 300

Northrop Corporation
1515 Rancho Conejo Blvd.
Newbury Park, CA 91320
Attn: Keith Hullinger

Rockwell International
North American Aircraft Division
P.O. Box 92098
Los Angeles, CA 90009
Attn: S.C. Mellin, Dept. 115, MA-09
Attn: E. Darkauskas, MB56

The Survice Engineering Co.
P.O. Box 693
Bel Air, MD 21014
Attn: James B. Foulk

Shock Hydrodynamics Division
Whittaker Corporation
4716 Vineland Ave.
No. Hollywood, CA 91602
Attn: Security Officer

Southwest Research Institute
P.O. Drawer 28510
San Antonio, TX 78284
Attn: P. H. Zabel, Div. 02

Comarco, Inc.
1417 N. Norma
Ridgecrest, CA 93555
Attn: G. Russell (2 copies)

Falcon Research & Development Co.
One American Drive
Buffalo, NY 14225
Attn: P.D.Pfohl

SECTION I

INTRODUCTION

The QKLOOK programs are a set of four FORTRAN programs which compute component, system, and total target vulnerable areas for Directed High Energy Weapons (DHEW). Vulnerable area is defined to be the product of presented area and probability of kill given a hit (P(K/H)) on that area. The target is described by a set of shot lines: parallel rays passed through the target, perpendicular to a specified viewing plane. These shot lines are generated from geometric target models by other computer programs such as FASTGEN or MAGIC. In addition to the shot line data, the user must supply detailed compone t information such as component code number, material type, density fac.or, initial operating temperature, and melt analysis type (melt-in-place or melt-removed), to be used in the penetration computations. The degree of component kill is determined by the depth to which the weapon is able to penetrate the component. Depths of penetration and associated P(K/H)'s for each critical component are specified by the user. QKLOOK then interpolates linearly between these values to determine actual component P(K/H) values (and hence vulnerable areas) at times specified by the user. The weapon is described simply by its flux density, which may (but need not) vary with time. The entire target is assumed to be flood-loaded with this flux density.

The four QKLOOK programs and their specific functions are:

CONMAG	- Converts shot line encounter data produced from a MAGIC target description to the format required by QKPK. (No conversion is necessary for a FASTGEN target description).
PRERD	- Checks the input data assembled for QKPK for possible errors, and produces several summary tables of that input data.
QKPK	- Simulates penetration along each shot line and computes penetration times for each encounter with a critical component.
PEAKAY	- Computes component, system, and total target vulnerable areas at specified time intervals using the penetration times computed by program QKPK.

The vulnerable areas generated by QKLOOK can be used in vulnerablility assessment and reduction studies, or as input to "end-game" computer programs where target probability of kill (Pk rather than P(K/H)) is determined.

ANALYST SECTIONS

Sections II, III, and IV are intended for use by the program analyst who must acquire a detailed understanding of the programs. Section II contains a set of conceptual flowcharts which provide a pictorial summary of the execution steps in the QKLOOK programs. The basic mathematical concepts employed in the QKLOOK programs are presented in Section III, the Mathematical Model. The Simulation Model, Section IV, provides a discussion of the individual FORTRAN statements in each of the QKLOOK program units. A dictionary of the FORTRAN variable names is included at the end of Section IV.

USER SECTIONS

Sections V, VI, and VII form the user portion of this manual. The information and data contained in these sections are primarily in tabular form so that they may be used as quick reference sources. Section V is used to present the formats, options, and definitions for all parameters used as input for the QKLOOK programs. The output for all of the programs is presented in Section VI. The binary output files are described in figures which list each output parameter as well as its units and definition. Each page of line printer output is also discussed and an example shown, including warning messages from the QKLOOK programs. Section VII is used to describe a sample problem. The figures for this section include copies of all formatted input and output. The sample problem is one that exercises many of the program features and user options. The user may also find the first part of Section II useful. A flowchart is given there showing the interaction of the various QKLOOK programs.

PROGRAM REQUIREMENTS/CONSTRAINTS

The largest QKLOOK program requires 22,197 (53265_8) words of memory on a UNIVAC 1110 computer. Peripheral requirements include one card reader, one line printer, and two mass storage devices. The programs in their present configurations have the following constraints:

1. No more than 498 components may be in the target model defined in the QKPK data deck unless array dimensions in the program are increased.

2. There must be no more than 100 encounters along any one shot line unless array dimensions in the program are increased.

3. No more than 10 systems of components may be defined unless array dimensions in the program are increased.

4. Maximum weapon intensity must be less than or equal to 60,000 watts/cm^2 in order for the penetration rate calculations to be valid.

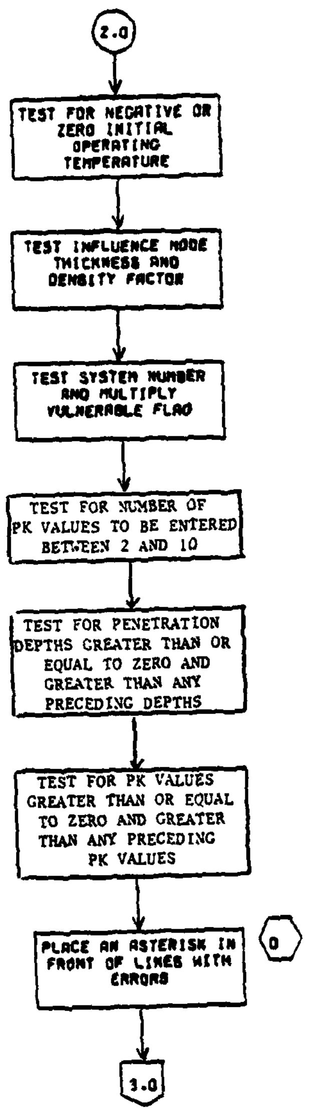

2.0
TEST FOR NEGATIVE OR ZERO INITIAL OPERATING TEMPERATURE
TEST INFLUENCE MODE THICKNESS AND DENSITY FACTOR
TEST SYSTEM NUMBER AND MULTIPLY VULNERABLE FLAG
TEST FOR NUMBER OF PK VALUES TO BE ENTERED BETWEEN 2 AND 10
TEST FOR PENETRATION DEPTHS GREATER THAN OR EQUAL TO ZERO AND GREATER THAN ANY PRECEDING DEPTHS
TEST FOR PK VALUES GREATER THAN OR EQUAL TO ZERO AND GREATER THAN ANY PRECEDING PK VALUES
PLACE AN ASTERISK IN FRONT OF LINES WITH ERRORS
D
3.0

FIGURE 2-5. QKLOOK: Program PRERD Conceptioal Flowchard (Page 2 of 3)

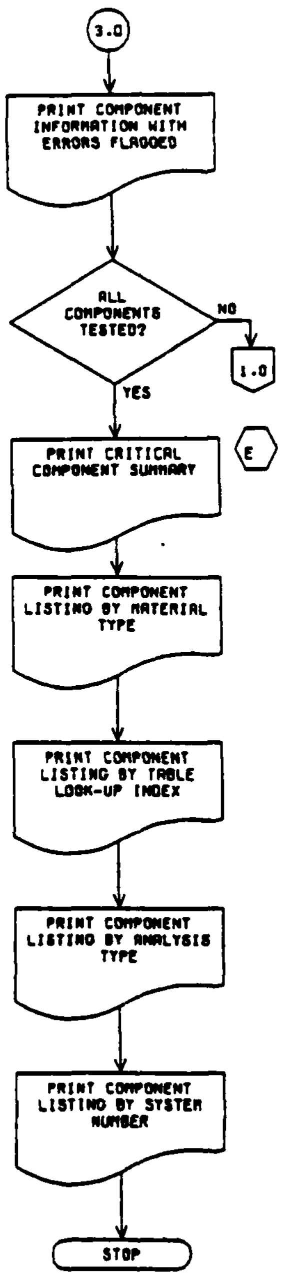

3.0
PRINT COMPONENT INFORMATION WITH ERRORS FLAGGED
ALL COMPONENTS TESTED?
NO
1.0
YES
E
PRINT CRITICAL COMPONENT SUMMARY
PRINT COMPONENT LISTING BY MATERIAL TYPE
PRINT COMPONENT LISTING BY TABLE LOOK-UP INDEX
PRINT COMPONENT LISTING BY ANALYSIS TYPE
PRINT COMPONENT LISTING BY SYSTEM NUMBER
STOP

FIGURE 2-5. QKLOOK: Program PRERD Conceptioal Flowchard (Page 3 of 3)

approximated by substituting the vaporization temperature for T* in Equation 3-11:

$$\alpha F \cong k \frac{(T_v - T_m)}{Z} \tag{3-13}$$

where

T_v = the component vaporization temperature, °C

The total heat absorbed during this type of encounter is approximated by multiplying Equation 3-13 by the component area and time:

$$Q = Atk \frac{(T_v - T_m)}{Z} \tag{3-14}$$

Similarly the heat required can be approximated by replacing T* with T_v in Equation 3-9

$$Q = AZ\rho \left[Cps\ (T_m - T_1) + \lambda + Cp\ell \left(\frac{T_v - T_m}{2} \right) \right] \tag{3-15}$$

The penetration rate equation for the melt-in-place analysis when the surface temperature exceeds T_v can be obtained by equating Equations 3-14 and 3-15, and solving for the rate, R where R = Z/t.

$$\frac{k}{Z} (T_v - T_m)\ At = AZ\rho \left[Cps\ (T_m - T_1) + \lambda + Cp\ell \left(\frac{T_v - T_m}{2} \right) \right]$$

$$\boxed{R = \frac{k \dfrac{(T_v - T_m)}{Z}}{\rho \left[Cps\ (T_m - T_1) + \lambda + Cp\ell \left(\dfrac{T_v - T_m}{2} \right) \right]}} \tag{3-16}$$

COMPONENT PROBABILITIES OF KILL

For each critical component along each shot line, penetration times are computed using the basic distance/rate equation:

$$t = \frac{d}{R} \tag{3-17}$$

where

t = the penetration time, seconds
d = the penetration distance, meters
R = the penetration rate, m/sec

The computed penetration times are:

t_N	= the time needed to penetrate from the beginning of the shot line to the depth necessary for a non-zero component Pk equal to Pk_N (N is in the range from one to one less than the number of Pk values specified), seconds
t_{N+1}	= the time needed to penetrate from the beginning of the shot line to the depth necessary for a component Pk equal to Pk_{N+1} (Pk_{N+1} is greater than or equal to Pk_N), seconds
t_L	= the time needed to penetrate from the beginning of the shot line through the line of sight thickness of the component, seconds

The component kill probability for a time, τ, is computed as follows ($N_{min} = 1$):

If $\tau < t_{N_{min}}$ or $t_L < t_{N_{min}}$ then $Pk = 0.0$

If $\tau \geq t_{N_{min}}$ and $t_L \geq t_{N_{min}}$ and $t_{N+1} \neq t_N$

then

$$Pk = \frac{(t_M - t_N)}{(t_{N+1} - t_N)} (Pk_{N+1} - Pk_N) + Pk_N \tag{3-18}$$

where

$t_M = \min(\tau, t_{N_{max}}, t_L)$

If $\tau \geq t_{N_{max}}$ and $t_L \geq t_{N_{max}}$, then $Pk = Pk_{N_{max}}$

COMPONENT PRESENTED AND VULNERABLE AREAS

The total presented area for a component can be computed very easily by counting the number of shot lines which intersect the component. If a shot line encounters a component one or more times, the presented area is assumed to fill the corresponding grid cell. Since there is only one shot line per grid cell, a component's presented area for one view can be expressed as:

$$A_p = N_S A_G \tag{3-19}$$

where

A_p = the total component presented area, ft^2

N_S = the number of shot lines encountering the component at least once, nondimensional

A_G = the grid cell area, ft^2

The vulnerable area of a component is defined as the product of its presented area and its kill probability.

$$A_V = Pk_T A_p \tag{3-20}$$

where

A_V = the total component vulnerable area, ft^2

Pk_T = the total component kill probability for the view, nondimensional

Substituting Equation 3-19 for the presented area, the vulnerable area is expressed as:

$$A_V = Pk_T\ N_S\ A_G \qquad (3\text{-}21)$$

In this problem, a component may be encountered in several grid cells.

Let

Pk_i = the component probability of kill for the *ith* shot line

Computing Pk_T as the average of the shot line kill probabilities:

$$Pk_T = \frac{\sum_{i=1}^{N_S} Pk_i}{N_S} \qquad (3\text{-}22)$$

then

$$Pk_T\ N_S = \sum_{i=1}^{N_S} Pk_i \qquad (3\text{-}23)$$

Substituting the right hand side of Equation 3-23 into Equation 3-21

$$\boxed{A_V = A_G \sum_{i=1}^{N_S} Pk_i} \qquad (3\text{-}24)$$

A component may also be encountered more than once along a shot line. The total P(K/H) for the component on the shot line is the sum of the probabilities of the exclusive events, as follows:

LIST OF ABBREVIATIONS AND SYMBOLS
(MATHEMATICAL MODEL)

Abbreviation or symbol	Equivalent in Simulation Model	Definition	Units
N_S	---	Number of shot lines encountering a component	---
N	---	Number of encounters for a component on the *jth* shot line	---
Pe_i	PK	Probability of kill for the *ith* encounter and no kill on a previous encounter	---
P_i	PK	Component probability of kill on the *ith* encounter	---
Pk_j	---	Component probability of kill from one shot line	---
Pk_T	---	Average of the probabilities of kill for all shot lines	---
Q	---	Heat absorbed by the component	joules
R	RATE	Penetration rate of a Directed High Energy Weapon against a component	m/sec
t	---	Penetration time (in Component Probabilities of Kill subsection); time duration of an encounter (in Penetration Rates subsection)	seconds
T*	TSTAR	Component outer surface temperature	^oC
T_1	TINIT	Component initial operating temperature	^oC

LIST OF ABBREVIATIONS AND SYMBOLS (MATHEMATICAL MODEL)

Abbreviation	Equivalent in Simulation Model	Definition	Units
T_m	TMLT	Component melting temperature	^oC
T_V	TVAP	Component vaporization temperature	^oC
t_L	PLS	Time required to penetrate from the beginning of a shot line to the depth necessary to perforate the component	seconds
t_M	PK3	Time needed to achieve the maximum Pk for a component in a time interval	seconds
t_N	PKTIME	Time required to penetrate from the beginning of a shot line to the depth associated with a Pk equal to Pk_N	seconds
Pk_N	PKVAL	Pk associated with penetration to a specified depth	---
Z	DP	Component actual line of sight thickness	meters
α	ALP	Component coupling coefficient, absorption factor	---
ΔE_{T_m}	---	Energy per unit mass needed to raise the temperature from T_1 to T_m	joules/kgm

SUBROUTINE BSRCH

This subroutine is used to find the link between a component number for an encounter and the component information arrays. This is done by executing a binary search of the component number array, ICOMP, which was sorted into ascending order in Subroutine FSORT. If the binary search can find no link, a default link is assigned.

The first set of statements

```
      SUBROUTINE BSRCH(ICOMP,NOCOMP,ITC)
      COMMON   /PROP/ ALP,RHO,CP,TMLT,XLAMBD,RATE,JCOMP,J,UP,XK,TVAP,
     $              CPL,ITEN,T
      COMMON   /LUNITS/ IRD,IWR,IIN,IOUT
      DIMENSION ICOMP(ITC)
```

is used to pass three arguments, ICOMP, NOCOMP, and ITC to this subroutine. ICOMP is the array being searched; NOCOMP is the number of values in this array; and ITC is the dimension of ICOMP. The statements are also used to specify two named commons which are used to transfer information into and out of this subroutine. In COMMON /PROP/, the variable JCOMP is the component number to be matched and the variable J is the subscript of the matching element in array ICOMP. The DIMENSION statement is used to declare the argument ICOMP to be an array.

The statements

```
      IBEG = 0
      IEND = NOCOMP + 1
   10 CONTINUE
        J = (IBEG + IEND) / 2
        IF (J .NE. IBEG) THEN
          IF (ICOMP(J) .LT. JCOMP) THEN
            IBEG = J
          ELSE IF (ICOMP(J) .GT. JCOMP) THEN
            IEND = J
          END IF
        ELSE
```

are used to perform the binary search. The algorithm begins by setting a pointer, IBEG, at the beginning of the array and another pointer, IEND, at the end of the array. An iterative procedure is used to set a third pointer, J, halfway between IBEG and IEND and to compare the Jth element with the desired value, JCOMP. The first block IF statement is false only if no match for JCOMP exists in the array ICOMP. In this case, the ELSE branch is executed. In all other cases, the block IF in the THEN branch is executed to compare the values JCOMP and ICOMP(J). If JCOMP is larger, the beginning pointer, IBEG, is moved to position J. If JCOMP is smaller, the end pointer, IEND, is moved to J. If the values are equal, the algorithm is finished.

The following statements

```
          IF (JCOMP .LT. 1000) THEN
            J = ITC - 1
          ELSE IF ((JCOMP .GE. 7000) .AND. (JCOMP .LE. 7999)) THEN
            J = ITC
          ELSE
            WRITE (IWR,100) JCOMP
            J = ITC
          END IF
          RETURN
        END IF
```

```
          IF (ICOMP(J) .NE. JCOMP) GO TO 10
          RETURN
      100 FORMAT (1X,I10,' WAS NOT FOUND IN LIST...DEFAULT TO 7075 AL')
          END
```

are used to allow a component number, JCOMP, to use default characteristics; i.e., the formatted input for the component information arrays does not need to include every component in all shot line encounters. An encounter component JCOMP with a number less than 1,000 and no match in the ICOMP array is assigned default data for a 2024 AL component. These default data for this type of component are stored at location ITC-1 in the component information arrays. Similarly, a component number JCOMP between the numbers 7000 and 7900 with no match in the ICOMP array is assigned default data (stored at location ITC in the component information array) for a 7075 AL component. If a component number, JCOMP, is in neither of the above ranges and has no match in the array ICOMP, the subroutine causes a warning message to be written. At this point, the default link ITC (7075 AL) is assigned. Control is then returned to the calling routine. The two END IF statements close the block IF statement assigning the default characteristics and the block IF checking whether a match for JCOMP may yet exist in the array ICOMP. If the algorithm has not yet completed, the GO TO statement branches to Statement 10 for another iteration.

are used to conclude the binary output file in QKLOOK format by writing a final binary record with the values of XX (= 999.0). The binary file is then rewound and the first record is overwritten. The new initial record is identical to the original one except that it includes the minimum and maximum coordinate values of all shot lines on the viewing plane. The purpose of the new initial record is to facilitate the use of a plotting program which would require the minimum and maximum values for scaling before the data can be plotted. Finally, the binary file on device IOUT is rewound in preparation for a run of Program QKPK.

The last set of statements

```
  180 FORMAT (F7.1,7X,F9.3,F9.3,I3,18X,F6.1,2X,F6.1)
  190 FORMAT (3(I4,F7.2,F5.1,I3,F7.2))
  200 FORMAT (2(I4,F7.2,7X,F6.1,I3,F7.2))
  210 FORMAT (1X,F6.1,F7.1,3X,F8.2,8X,I3)
  220 FORMAT (I5,10A6)
  225 FORMAT (15H1NO. OF VIEWS =,I5/1H0,10A6)
  230 FORMAT (2(5X,E15.8),30X,E10.3)
  235 FORMAT (10H0AZIMUTH =,F10.2,13H  ELEVATION =,F10.2,
     1  13H  GRID SIZE =,F10.2)
  240 FORMAT (7H0YMAX =,F10.2,8H  YMIN =,F10.2/7H0ZMAX =,F10.2,
     2  8H  ZMIN =,F10.2)
  245 FORMAT (1H1)
 1000 FORMAT (1H1,69HTHE FOLLOWING IS AN ECHO OF THE DATA WRITTEN ON THE
     1 MAGIC OUTPUT FILE//1H0,30HA, EL, GRID, LL, X, X, X, X, R/
     2  1H ,3F10.3,I10,5F10.2)
 1001 FORMAT (1H0,69HAX(I), AY(I), AZ(I), ISHI(I), IECO(I), ITH(I), IIB(
     1I),IOB(I), I=1,170/(1H ,3F10.3,5I10))
 1002 FORMAT (1H0,30HXX, XX, XX, II, XX, XX, XX, XX, XX/1H ,3F10.3,I10,
     1 5F10.3)
 1003 FORMAT (1H0,42HAZM, ELEV, GRID, LL, YMX, YMN, ZMX, ZMN, R/
     1  1H ,3F10.3,I10,5F10.3/1H ,5X,73H(THE RECORD ABOVE ACTUALLY OVER-
     2WRITES THE VERY FIRST RECORD ON THE FILE)/1H0,32HEND OF ECHO OF MA
     1GIC OUTPUT FILE/1H1)
      END
```

is used to define all formats used for input and output within the program and to indicate the end of this program.

SUBROUTINE FSORT

This subroutine is invoked by two programs in the QKLOOK set, Programs PRERD and QKPK. It is used to sort the component information arrays into ascending order by component number. ICOMP is the array of component numbers and is the basis for the sorting process while NOCOMP is the number of components in the array ICOMP. With the exception of ITC and PKNBR, which are used to dimension the passed arrays, the remaining arguments are other component information arrays. These information arrays correspond to the component number array by subscripts; therefore, a switch in this routine will require switches in each of the 12 corresponding arrays. Subroutine FSORT is used to perform a diminishing increment sort which requires a large number of comparisons but a relatively small number of switches, making this algorithm very efficient for this type of problem.

The first set of statements

```
 SUBROUTINE FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,
$                DEPTH,PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
 INTEGER IANA  (ITC),
$        ICOMP (ITC),
$        IFG   (ITC),
$        ITAB  (ITC),
$        IU    (ITC),
$        IY    (ITC),
$        MAT   (ITC),
$        NOPNTS(ITC),
$        PKNBR
 REAL    BUFFR1(50),
$        BUFFR2(50),
$        DEPTH (PKNBR,ITC),
$        PKVAL (PKNBR,ITC),
$        RHOF  (ITC),
$        THINFL(ITC),
$        TINIT (ITC)
```

is used to transfer 16 arguments to and from the calling programs, and to declare 13 of these arguments to be arrays.

The statements

```
    M = NOCOMP
 10 CONTINUE
    M = M / 2
    IF (M .LT. 1) GO TO 50
```

are used to compute and test the increment size M. Initially, M is half of the array length. After each of the M sublists is sorted, the routine returns to Statement 10. The increment size M is halved again and the sublist is sorted. The final sort is executed when M equals 1; then, the above IF statement causes a branch to Statement 50, the end of the subroutine.

The next statements

```
    K = NOCOMP - M
    DO 40 J = 1,K
```

are used to compute K, the number of steps required to sort the M sublists, and to initiate the execution of a DO loop for every step.

The statements

```
            I = J
   20       CONTINUE
              IM = I + M
              IF (ICOMP(I) .LE. ICOMP(IM)) GO TO 40
```

are used to perform a comparison between two elements in a sublist. The computation for IM is used to indicate that each sublist consists of all array elements at intervals equal to the increment size M. If the comparison shows that these two elements are already in ascending order, the routine branches to Statement 40, the end of the DO loop; otherwise, the two array elements and all corresponding array elements must be interchanged.

These statements

```
              N0 = ICOMP(I)
              N1 = MAT(I)
              N2 = IFG(I)
              N3 = ITAB(I)
              N4 = IANA(I)
              N5 = IY(I)
              N6 = IU(I)
              N7 = NOPNTS(I)
              S1 = TINIT(I)
              DO 30 MM = 1,PKNBR
                BUFFR1(MM) = DEPTH(MM,I)
                BUFFR2(MM) = PKVAL(MM,I)
   30         CONTINUE
              S4 = THINFL(I)
              S5 = RHOF(I)
              ICOMP(I) = ICOMP(IM)
              MAT(I) = MAT(IM)
              IFG(I) = IFG(IM)
              ITAB(I) = ITAB(IM)
              IANA(I) = IANA(IM)
              IY(I) = IY(IM)
              IU(I) = IU(IM)
              NOPNTS(I) = NOPNTS(IM)
              TINIT(I) = TINIT(IM)
              DO 34 MM = 1,PKNBR
                DEPTH(MM,I) = DEPTH(MM,IM)
                PKVAL(MM,I) = PKVAL(MM,IM)
   34         CONTINUE
              THINFL(I) = THINFL(IM)
              RHOF(I) = RHOF(IM)
              ICOMP(IM) = N0
              MAT(IM) = N1
              IFG(IM) = N2
              ITAB(IM) = N3
              IANA(IM) = N4
              IY(IM) = N5
              IU(IM) = N6
              NOPNTS(IM) = N7
              TINIT(IM) = S1
              DO 38 MM = 1,PKNBR
                DEPTH(MM,IM) = BUFFR1(MM)
                PKVAL(MM,IM) = BUFFR2(MM)
   38         CONTINUE
              THINFL(IM) = S4
              RHOF(IM) = S5
```

are used to switch the two compared elements along with all elements from the 12 other arrays corresponding to them.

The statements

```
      I = I - M
      IF (I .GE. 1) GO TO 20
```

are used to finish the sort of the sublist. After the switch of two sublist elements in the previous step, I is decremented to the next element lower in the sublist. The IF statent is used to branch back to Statement 20 where the two lower elements of the sublist are sorted. The IF statement does not branch back when I is already at the bottom of the sublist.

The statements

```
40    CONTINUE
      GO TO 10
```

are used to end the DO loop and cause a branch to Statement 10. The DO loop is completed when all of the M sublists are sorted. The branch to Statement 10 is used to halve the increment size and repeat the above steps.

The statements

```
50 CONTINUE
   RETURN
   END
```

are executed only after the increment size M is less than one, which means that the array ICOMP has been sorted into one sublist. These statements are used to return control to the calling subroutine and to end this program unit.

PROGRAM PEAKAY

This program is the fourth in the series of QKLOOK programs. It is used to compute component, system, and total target kill probabilities and vulnerable areas. The encounter penetration times, which are read from the binary file generated by Program QKPK, are used to compute the kill probabilities in Subroutine PENT, the only subroutine in this program. After the component and system Pk's are known, the vulnerable area computations are done in Program PEAKAY with the assumption that the presented area for each encounter equals the grid size. The printed output includes presented areas and vulnerable areas for individual components, systems of components, and total target. These data can be computed for up to 10 time intervals specified by the user in the formatted input. A greater number of time intervals may be obtained by using multiple runs with the same binary input file. A copy of all input, the shot line Pk's, and the formatted output are also produced on four binary output files.

The first group of statements

```
      INTEGER   AVFLAG
      COMMON    R1(10,200),R3(200),I1(200),PFTIME(10,100),
     $          PKVAL(10,100),NOPNTS(500),PLS(100),
     $          NAME(100),IST,NSHT,NENC,ICOMP(500),SHR(12,200),
     $          NOCOMP,COMPAV(500,10),GRID,TIMES(10),NTIME,IY(500),
     $          PMULT(10,10),IU(500),PAREA(500)
      COMMON   /ONE/ IFG(500),PRNT(4,500),AVFLAG
      COMMON   /LUNITS/ IRD,IWR,IIN,IOUT
      DIMENSION PKNK(500),TVA(10),TVAM(10),ISET(10),SAREA(10),
     $          FTIM(25),FXCM(25)
```

is used to declare AVFLAG as integer, to allocate common storage areas, and to declare array dimensions.

The statements

```
      DATA      IRD,IWR,IIN,IOUT1,IOUT2,IOUT3,IOUT4 /7,8,2,10,11,12,13/
      DATA      PMULT /100*0./
      DATA      IST,NUM,NSHT /3*0/
      DATA      PAREA /500*0./
      DATA      (COMPAV(I,1),I=1,500) /500*0./
      DATA      (COMPAV(I,2),I=1,500) /500*0./
      DATA      (COMPAV(I,3),I=1,500) /500*0./
      DATA      (COMPAV(I,4),I=1,500) /500*0./
      DATA      (COMPAV(I,5),I=1,500) /500*0./
      DATA      (COMPAV(I,6),I=1,500) /500*0./
      DATA      (COMPAV(I,7),I=1,500) /500*0./
      DATA      (COMPAV(I,8),I=1,500) /500*0./
      DATA      (COMPAV(I,9),I=1,500) /500*0./
      DATA      (COMPAV(I,10),I=1,500) /500*0./
      DATA      (PRNT(1,I),I=1,500) /500*0./
      DATA      (PRNT(2,I),I=1,500) /500*0./
      DATA      (PRNT(3,I),I=1,500) /500*0./
      DATA      (PRNT(4,I),I=1,500) /500*0./
      DATA      ITOTL /0/
      DATA      TVA /10*0./
      DATA      SAREA /10*0./
      DATA      ITC /500/
      DATA      PCV /90./
      DATA      SQFSUM /0.09290304/
```

represent DATA statements, which are used to initialize several variables. The variables in the first DATA statement are the input and output device numbers. The last three DATA statements are used to initialize the dimension of the component arrays, ITC, the percent of vulnerable area, PCV, and a factor for conversion from square feet to square meters, SQFSQM. The rest of the variables in the DATA statements are initialized equal to 0.00

The statements

```
OPEN (IRD,FILE='PKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
OPEN (IWR,FILE='PKPRINT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
OPEN (IIN,FILE='QKPKTAPEOUT',FORM='UNFORMATTED',RECFM='VARIABLE')
OPEN (IOUT1,FILE='PKTAPEOUT1',FORM='UNFORMATTED',RECFM='VARIABLE')
OPEN (IOUT2,FILE='PKTAPEOUT2',FORM='UNFORMATTED',RECFM='VARIABLE')
OPEN (IOUT3,FILE='PKTAPEOUT3',FORM='UNFORMATTED',RECFM='VARIABLE')
OPEN (IOUT4,FILE='PKTAPEOUT4',FORM='UNFORMATTED',RECFM='VARIABLE')
```

are used to connect logical units to the input and output files and to establish the connection properties between each unit/file pair.

The statements

```
 REWIND IIN
 READ (IRD,1003) NTIME,AVFLAG
 READ (IRD,1001) (TIMES(J),J=1,NTIME)
 READ (IIN) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCUMP
 READ (IIN) (ICOMP(I),IFG(I),IY(I),IU(I),NOPNTS(I),
$  (PKVAL(J,I),J=1,NOPNTS(I)),I=1,NOCOMP),IRVRS,IFMAX,
$  (FTIM(I),FXCM(I),I=1,IFMAX)
```

are used to read the formatted input file and the first two records of the binary file from Program QKPK. The REWIND statement is used insure that the first binary read will start at the file origin. The two formatted READ statements are used to read the number of time intervals, the flag determining whether component vulnerabilities are to be calculated as true or as incremental, and the times for each interval. This is the only formatted input required for this program. The two binary READ statements are used to read the first two records of the binary input file. These data include the viewing grid description, the component information arrays, and the flux distribution table.

The next statements

```
       IF (ITC .GE. NUCOMP+2) THEN
         WRITE (IWR,4) NTIME,(TIMES(J),J=1,NTIME)
         RVRS = IRVRS
         WRITE (IWR,8) RVRS
         WRITE (IWR,800) IFMAX
         TE = 0.
         DO 820 I = 1,IFMAX
           TB = TE
           TE = FTIM(I)
           WRITE (IWR,810) TB,TE,FXCM(I)
  820    CONTINUE
         TB = TE
         TE = 1.E30
         WRITE (IWR,810) TB,TE,FXCM(IFMAX)
```

are used to test the number of components and print the first page of formatted output. The block IF statement is used to insure that the number of components does not exceed array dimensions. If this fatal error occurs, the program will execute the ELSE branch of the block IF. Note that Program QKPK also tests and aborts on the same error. The first WRITE statement is used to print the number of time intervals and the time values for which Pk's and vulnerable areas will be computed. This is followed by two WRITE statements that are used to print the reverse flag and the number of points in the flux distribution. The rest of the statements are used to print the flux distribution table. The DO loop is used to print flux array values with corresponding begin times and end times. The final WRITE statement is used to print the final line of the flux table indicating that the last flux level applies to all times greater than the last end time.

The statements

```
      WRITE (IOUT1) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
      WRITE (IOUT1) (ICOMP(I),IFG(I),IY(I),IU(I),I=1,NOCOMP),IRVRS,
     $  IFMAX,(FTIM(I),FXCH(I),I=1,IFMAX)
   10 CONTINUE
        READ (IIN) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
        WRITE (IOUT1) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
      IF (I1(200) .NE. 9999) GO TO 10
```

are used to save a copy of the binary input data from Program QKPK on the first binary output file. The first two binary WRITE statements are used to write the data from the first two records of binary input. The statements following Statement 10 are used to iteratively read and then write the binary records. The IF statement is used to test for the end of view flag and to continue to loop until the last binary record has been echoed. Note that data written on device IOUT1 are in the identical order as data on device IIN.

The statements

```
      END FILE IOUT1
      REWIND IIN
      GRID = GRID * GRID / 144.
```

are used to place an end of file mark on the first binary output file, to rewind the binary input file, and to convert the variable GRID from a length to an area expressed in square feet.

The statements

```
      READ (IIN)
      READ (IIN)
      MAXTIM = NTIME + 2
```

are used to skip the first two records of the binary input file and compute the variable MAXTIM, which is a dimension size for the SHW array. By using the two binary READ statements to pass two records, the binarv input file is now at the start of the shot line penetration times computed by Program QKPK.

The statements

```
  100 CONTINUE
        READ (IIN) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
        DO 110 J = 1,200
          IF (I1(J) .EQ. 9999) GO TO 5000
```

are the starting point for the time file processing loop. The binary READ is used to read and store one binary record which fills three arrays. The DO statement is used to initiate a DO loop which iterates 200 times to process each set of array elements. The IF statement is used to branch out of the time file processing loop when an end of view flag is detected. Since the end of the DO loop is followed by the statement GO TO 100, Statement 100 is used as a return point so that another record of binary input may be read and processed.

The statements

```
                  IF (IST .LE. 0) THEN
                    ITOTL = ITOTL + 1
                    NENC = I1(J)
                    NSHT = NSHT + 1
                    SHW(1,NSHT) = R1(1,J)
                    SHW(2,NSHT) = R1(2,J)
                    IF (NENC .NE. 0) THEN
                      IST = NSHT
                      NUM = 0
```

are used to store the initial data for each new shot line on the viewing plane. The block IF statement is used to test the shot line index, IST. The index IST equals 0 when a new shot line is being defined. When the IF statement detects a new shot line, the preceding statements are used. The variable ITOTL, which counts the number of shot lines in the view, is incremented by using the first assignment statement. The number of encounters on the shot line is assigned to NENC and the subscript NSHT is incremented by using the next two assignment statements. The first two rows of the array SHW are used to store shot line coordinates on the viewing plane. The second block IF statement checks if there is at least one critical encounter on the shot line. If so, the next two statements are executed to assign the shot line numbers to IST and to initialize the subscript NUM.

The statements

```
                  ELSE
                    DO 112 MM = 3,MAXTIM
                      SHW(MM,NSHT) = -9.
      112           CONTINUE
                  END IF
```

are executed only when a shot line has no critical encounters. The DO loop is used to fill the rest of the SHW array with the value -9.0, designated as a flag to indicate no critical encounters. The END IF statement closes the block IF.

The statements

```
                  ELSE
                    NUM = NUM + 1
                    DO 114 K = 1,10
                      PKTIME(K,NUM) = R1(K,J)
      114           CONTINUE
                    PLS(NUM) = R3(J)
                    NAME(NUM) = I1(J)
```

can be executed only when processing a set of binary input data that contains penetration times for an encounter (i.e., not the start of a new shot line). The first assignment statement is used to increment the variable NUM and the remaining assignment statements are used to store the eleven penetration times as well as the component number location in the PKTIME, PLS, and NAME arrays, indexed by the encounter number NUM.

The next statements

```
                IF (NUM .GE. NENC) THEN
                  DO 115 I = 1,10
                    ISET(I) = 0
115               CONTINUE
                  DO 118 I = 1,NENC
                    II = NAME(I)
                    LL = IY(II)
                    IF (LL .NE. 0) ISET(LL) = 1
                    IF (I .NE. NENC) THEN
                      DO 116 I3 = I+1,NENC
                        IF (II .EQ. NAME(I3)) GO TO 118
116                   CONTINUE
                    END IF
                    PAREA(II) = PAREA(II) + 1.
118               CONTINUE
```

are executed only after the data for every encounter on a shot line have been stored. The first DO loop is used to assign zeros to the system encounter array ISET. This is followed by a DO loop which is used to increment the presented area array PAREA for each component on this shot line and to assign a one to the ISET array for every system represented on this shot line. The first two assignment statements in the loop are used to assign the component location to II and the system number to LL. The logical IF statement is used to allocate a one to the ISET array if the encountered component is in a system. The last assignment statement in the loop is used to increment the PAREA array for the encountered component. The block IF with its DO loop is used to insure that the presented area for a component is incremented only once even if the component is in more than one encounter on a shot line.

The statements

```
                  DO 119 I = 1,10
                    SAREA(I) = SAREA(I) + ISET(I)
119               CONTINUE
```

are used to accumulate the number of shot lines that encounter each system into the SAREA array. When all shot lines for a view have been processed, the SAREA array contains the number of shot lines that have an encounter with a component in each system.

The next statements

```
                  CALL PENT
                  IST = 0
```

are used to invoke Subroutine PENT and reset the shot line index IST. Subroutine PENT is used to perform the shot line penetration computations for component, system, and total shot line kill probabilities at the time intervals specified by the user's formatted input. These data are transferred using the common storage areas.

The statements

```
                  DO 107 II = 1,NTIME
                    TVA(II) = TVA(II) + SHW(II+2,NSHT)
107               CONTINUE
```

are used to store the cumulative kill probabilities for all time intervals. After all shot lines have been processed, the array TVA contains the cumulative kill probabilities for a shot line at each time interval from the first to the NTIME intervals. These data are used later to compute the total target vulnerable area at each time interval by multiplying the kill probability by the presented area.

The next statements

```
                  END IF
                END IF
                IF (NSHT .EQ. 200) THEN
                  WRITE (IOUT2) ((SHW(II,MM),II=1,12),MM=1,200)
                  NSHT=0
                END IF
    110       CONTINUE
            GO TO 100
```

are used to end the time file processing loop. The first END IF statement closes the IF block used when all encounters on a shot line have been processed. The second END IF closes the IF block that tests whether a new shot line is being defined. The block IF statement is used to execute the binary WRITE statement when the SHW array has data from 200 shot lines. The execution of the binary WRITE statement is followed by an assignment statement which is used to reset NSHT, the number of shot lines in array SHW, to zero. If the array is not full, the block IF is bypassed and execution continues at Statement 110. This statement is used to end the time file processing DO loop. After the DO loop has been executed for each of the 200 elements in the binary input arrays, the GO TO statement is used to return to Statement 100 to read and process a new set of binary input.

The statements

```
    5000   CONTINUE
           NSHT = NSHT + 1
           DO 5010 K = 3,MAXTIM
             SHW(K,NSHT) = -9999.
    5010   CONTINUE
           WRITE (IOUT2) ((SHW(II,MM),II=1,12),MM=1,200)
           END FILE IOUT2
```

are executed only after the time file processing loop has detected the end of view flag. The assignment statement and DO loop are used to assign an end of view flag equal to -9999.0 in the next row of the SHW array. The binary WRITE statement is used to send the last set of shot line Pk data to the binary file on device IOUT2. The ENDFILE statement is used to put an end of file mark on the binary output file.

The statements

```
           WRITE (IWW,7001) PCV
           JQQ = 0
           FKV = PCV * 0.01
```

are used to begin the next page of PEAKAY printed output. Execution of the WRITE statement prints the heading for the page of output containing the time intervals during which the vulnerable areas reach 90 percent of the presented area. The two assignment statements are used to initialize the counter JQQ and to convert the required percent of vulnerable area PCV to decimal form. (The value of 90 percent for the variable PCV is assigned in a DATA statement. If another percentage is desired, a change of value in the DATA statement is the only action required).

The next statements

```
         DO 8000 MQ = 1,NOCOMP
           IF (IFG(MQ) .NE. 0) THEN
             IF(PRNT(3,MQ) .LE. 0.) THEN
               JQQ = JQQ + 1
               PKNK(JQQ) = ICOMP(MQ)
             END IF
```

are used to begin a loop which determines the time interval during which each component's vulnerable area reaches 90 percent of its presented area. The DO statement is used to initiate the loop which iterates for every component. The first block IF statement is used to exclude noncritical components from this summary by passing control to the end of the loop if the component's criticality flag is not set. The second block IF statement is used to determine the components which have a zero probability of kill by testing the third row of the PRNT array. The statements in the IF block are used to count and save the component numbers for components which have a zero probability of kill. The variable JQQ is used to count these components, and their component numbers are saved in array PKNK.

The statements

```
             IF (PAREA(MQ) .GT. 0.) THEN
               DO 6010 K = 1,NTIME
                 IF (COMPAV(MQ,K)/PAREA(MQ) .GE. FRV) THEN
                   IF (K .GT. 1) THEN
                     TB = TIMES(K-1)
                   ELSE
                     TB = 0
                   END IF
                   WRITE (IWR,7002) ICOMP(MQ),(PRNT(JJ,MQ),JJ=1,4),TB,
     $               TIMES(K)
                   GO TO 8000
                 END IF
 6010          CONTINUE
             END IF
             WRITE (IWR,7002) ICOMP(MQ),(PRNT(JJ,MQ),JJ=1,4)
           END IF
 8000    CONTINUE
```

are used to select the time interval for which each component's vulnerable area reaches 90 percent of its presented area and to print that information. The first IF block is used to bypass components with no presented area; i.e., no shot line has an encounter with the component. For components with a presented area, the DO loop is used to test the component vulnerable area at each time interval. The next IF block is used only for components whose vulnerable area reaches 90 percent. The block contains another IF block to determine the time interval during which the vulnerable area reached 90 percent of the presented area. The WRITE statement prints component encounter information contained in the PRNT array as well as the time interval previously determined. The GO TO statement then branches to the end of the component loop. The END IF statement following Statement 6010 closes the IF block for components having a presented area. The following WRITE statement is executed for components whose vulnerable area does not reach 90 percent of the presented area as well as for those which have no presented area. Statement 8000 is used to end the DO loop executed for every component.

The next statements

```
      DO 305 J = 1,NOCOMP
        PAREA(J) = PAREA(J) * GRID
        DO 300 K = 1,NTIME
          COMPAV(J,K) = COMPAV(J,K) * GRID
300     CONTINUE
305   CONTINUE
```

use two nested DO loops to compute component presented and vulnerable areas. The presented area, A_p, is computed using the expression:

$$\boxed{A_p = N_S A_G} \tag{3-19}$$

The vulnerable area is computed using the expression:

$$\boxed{A_V = A_G \sum_{i=1}^{N_S} Pk_i} \tag{3-20}$$

The sum of shot line Pk's was computed in Subroutine PENT and stored in the COMPAV array. The outer DO loop is used to iterate for every component and the inner loop is used to iterate for every time interval.

The statements

```
      DO 315 K = 1,NTIME
        TVA(K) = TVA(K) * GRID
        TVAM(K) = TVA(K) * SQFSUM
        DO 310 J = 1,10
          PMULT(J,K) = PMULT(J,K) * GRID
310     CONTINUE
315   CONTINUE
```

are used to convert the total target and system kill probabilities to vulnerable areas. The outer DO loop is used to iterate for each time interval. The first assignment statement is used to compute the total vulnerable area as the product of the total Pk and the specified grid area. The second assignment statement is used to compute the total vulnerable area in square meters and store it in the TVAM array. The inner DO loop is used to iterate for each of 10 possible systems. The last assignment statement is used to compute the system vulnerable area by multiplying the system Pk and the grid area expressed in square feet.

The statements

```
        IF (AVFLAG .EQ. 1) THEN
          WRITE (IWR,1004) (TIMES(K),K=1,NTIME)
        ELSE
          WRITE (IWR,1007) (TIMES(K),K=1,NTIME)
        END IF
        WRITE (IOUT3) AZ,EL,IFMAX,(FTIM(I),FXCM(I),I=1,IFMAX),NVRS,
     $    NOCOMP,NTIME,(TIMES(I),I=1,NTIME)
```

```
      DO 320 J = 1,NOCOMP
        IF (IFG(J) .NE. 0) THEN
          WRITE (IWR,1005) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
          WRITE (IOUT3) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
        END IF
320   CONTINUE
      END FILE IOUT3
      TOTL = ITOTL*GRID
      WRITE (IWR,7005) TOTL,ITOTL
      WRITE (IWR,7006) (TVA(K),K=1,NTIME)
```

are used to write the third page of PEAKAY printed output and the third binary file. The first IF block is used to print the page heading and time increments, depending on whether true component vulnerabliities (AVFLAG = 1) or incremental component vulnerabilities (AVFLAG = 0) are calculated. The first binary WRITE statement is used to write the attack angles, the flux table, the reverse flag, the number of components, and the time intervals on the binary output file. The DO loop is used to print the component number, the presented area, and the vulnerable area at each time increment for every critical component. Two WRITE statements are used to write these data on both output devices. The IF block is used to bypass the WRITE statements for noncritical components. The ENDFILE statement is used to write an end of file after the third binary file. The assignment statement is used to compute the total target presented area as the product of the total number of shot lines and the grid size. The last two formatted WRITE statements are used to print the total presented area, the total number of shot lines, and the total target vulnerable area at each time increment.

The next statements

```
      WRITE (IWR,1008) (TIMES(K),K=1,NTIME)
      DO 325 K = 1,10
        SAREA(K) = SAREA(K) * GRID
        WRITE (IOUT4) SAREA(K),K,(PMULT(K,J),J=1,NTIME)
        WRITE (IWR,1005) K,SAREA(K),(PMULT(K,J),J=1,NTIME)
325   CONTINUE
```

are used to start the fourth binary output file and to print the fourth page of PEAKAY formatted output. The first WRITE statement is used to print the heading and time increments for the new page of formatted output. The DO loop is executed for each of ten possible systems. The assignment statement is used to compute the system presented area by multiplying the number of shot lines by the grid area. This is followed by two WRITE statements which write the system number, system presented area, and the system vulnerable area for each time increment on both output devices. If a system number is not used in the target model, the system number is printed with 0.0 values for the presented and vulnerable areas. It should be noted that a multiply vulnerable component does not contribute to any system vulnerable area.

Execution of the binary WRITE statement

```
      WRITE (IOUT4) JQQ,(PKNK(NCN),NCN=1,JQQ)
```

records the component number of any component with a kill probability equal to zero on the fourth binary output file.

The next group of statements

```
      DO 360 J = 1,NOCOMP
        PAREA(J) = PAREA(J) * SQFSQM
        DO 350 K = 1,NTIME
          COMPAV(J,K) = COMPAV(J,K) * SQFSQM
350     CONTINUE
360   CONTINUE
      IF (AVFLAG .EQ. 1) THEN
        WRITE (IWR,1006) (TIMES(K),K=1,NTIME)
      ELSE
        WRITE (IWR,1011) (TIMES(K),K=1,NTIME)
      END IF
      DO 370 J = 1,NOCOMP
        IF (IFG(J) .NE. 0)
     $    WRITE (IWR,1005) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
370   CONTINUE
      TOTL = TOTL * SQFSQM
      WRITE (IWR,7007) TOTL,ITOTL
      WRITE (IWR,7008) (TVAM(K),K=1,NTIME)
```

is used to print component presented area and vulnerable area table values expressed in square meters. The first two nested DO loops are used to convert the presented areas and the vulnerable areas from square feet to square meters for every component and every time increment. The IF block prints the page heading and time increments, depending on whether true component vulnerabilities (AVFLAG = 1) or incremental component vulnerabilities (AVFLAG = 0) are calculated. The second DO loop is used to print the presented area and vulnerable area tables for all critical components. Noncritical components are excluded from the table by using the IF statement inside the loop. The last three statements are used to convert the total presented area to square meters and to print the total target vulnerable areas.

Then the statements

```
      WRITE (IWR,1009) (TIMES(K),K=1,NTIME)
      DO 400 K = 1,10
        SAREA(K) = SAREA(K) * SQFSQM
        DO 380 J = 1,NTIME
          PMULT(K,J) = PMULT(K,J) * SQFSQM
380     CONTINUE
        WRITE (IOUT4) SAREA(K),K,(PMULT(K,J),J=1,NTIME)
        WRITE (IWR,1005) K,SAREA(K),(PMULT(K,J),J=1,NTIME)
400   CONTINUE
```

are used to form two nested DO loops used to print the system presented area and vulnerable area tables after converting all values to square meters. The outer DO loop is used to iterate for each of the ten possible systems. After the system presented area is converted, the inner loop is used to convert the system vulnerable areas at each time increment. The last two WRITE statements each contain an implied DO loop, and are used to write the system number, presented area, and vulnerable area at each time increment. Statement 400 is used to end the system DO loop. These statements are used to write a record on each output device for the 10 system numbers. If a system number is not used in the target model, a record is printed with zeros for the presented and vulnerable areas.

The statements

```
      WRITE (IWR,7011)
      DO 330 J = 1,NOCOMP
        IF ((IFG(J) .NE. 0) .AND. (PAREA(J) .LE. 0.))
     $    WRITE (IWR,7012) ICOMP(J)
```

```
        330   CONTINUE
```

are used to print the seventh page of PEAKAY formatted output which consists of a list of critical components whose presented areas equal zero. The first WRITE statement is used to print the page heading. The DO loop is used to iterate for every component. The IF statement is used to print the component number for critical components with zero presented area. Other components are bypassed.

The next set of statements

```
              WRITE (IWR,7013)
              DO 340 J = 1,NOCOMP
                IF ((IFG(J) .NE. 0) .AND. (PRNT(3,J) .LE. 0.)) THEN
                  ASQF = PAREA(J)/SQFSUM
                  WRITE (IWR,1010) ICOMP(J),ASQF,PAREA(J)
                END IF
        340   CONTINUE
```

is used to print the last page of formatted output, a list of critical components with zero vulnerable area. The first WRITE statement is used to print the page heading. The DO statement is used to initiate a loop to check every component. The IF block checks for critical components whose vulnerable area is zero. For those components, the presented area is converted back to square feet by executing the assignment statement. The last WRITE statement is used to print the component number, and the presented area expressed in both square feet and square meters for any component with zero vulnerable area.

The statement

```
              WRITE (IOUT4) (TVA(I),TVAM(I),I=1,NTIME)
```

is used to write the total target vulnerable areas in square feet and square meters at each time increment on the last binary output record.

The statements

```
              ELSE
                WRITE (IWR,7014)
              END IF
              STOP
```

are the final executable statements in the program. The ELSE branch of the block IF is executed only if the number of components is too large for the array dimensions. The write statement is used to print a fatal error message on the formatted output file, and the STOP statement is used to halt program execution. Program QKPK tests and aborts on this same error.

The last group of statements

```
    4 FORMAT (' NTIME=',I5,' STEPS'/10X,' TIME STEPS ARE:',10F10.3)
    6 FORMAT (10X,' INVRS=',F5.0)
  800 FORMAT (' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5/
     $  ' BEGIN TIME',5X,'END TIME',7X,'FLUX')
  810 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
 1001 FORMAT (10F6.2)
 1002 FORMAT (5(3F6.2,2X))
 1003 FORMAT (2I5)
```

```
1004 FORMAT ('1 PRESENTED AREA AND TRUE COMPONENT VULERABLE AREAS ',
     $  '(SQ. FEET) PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
1005 FORMAT (I5,F10.5,5X,10F10.5)
1006 FORMAT ('1 PRESENTED AREA AND TRUE COMPONENT VULERABLE AREAS ',
     $  '(SQ. METERS) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
1007 FORMAT ('1 PRESENTED AREA AND INCREMENTAL COMPONENT VULERABLE ',
     $  'AREAS (SQ. FEET) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
1008 FORMAT ('1 PRESENTED AREA AND SYSTEM VULERABLE AREAS (SQ. FEET) ',
     $  'PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
1009 FORMAT ('1 PRESENTED AREA AND SYSTEM VULERABLE AREAS (SQ. METERS)'
     $  ' PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
1010 FORMAT (1X,I6,9X,2E12.5)
1011 FORMAT ('1 PRESENTED AREA AND INCREMENTAL COMPONENT VULERABLE ',
     $  'AREAS (SQ. METERS) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
7001 FORMAT('1',5X,'COMP. NO    Y-COORD.    Z-COORD.     MAX. PK     ',
     $  ' TIME',5X,'TIME INTERVAL DURING WHICH VULERABLE AREA REACHES',
     $  F4.0,'%'/66X,'OF THE PRESENTED AREA'/)
7002 FORMAT (I11,2X,2E12.4,F11.4,F12.4,10X,F12.4,' TO',F12.4,' SECONDS'
7005 FORMAT (' TOTAL TARGET PRESENTED AREA (SQ. FEET) =',F10.5,
     $  5X,'(',I6,' TOTAL SHOTLINES)')
7006 FORMAT (' TOTAL TARGET VULNERABLE AREA (SQ. FEET) PER ',
     $  'TIME INCREMENT'//(20X,10F10.5))
7007 FORMAT (' TOTAL TARGET PRESENTED AREA (SQ. METERS) =',F10.5,
     $  5X,'(',I6,' TOTAL SHOTLINES)')
7008 FORMAT (' TOTAL TARGET VULNERABLE AREA (SQ. METERS) PER ',
     $  'TIME INCREMENT'//(20X,10F10.5))
7011 FORMAT ('1 CRITICAL COMPONENTS WHOSE PRESENTED AREA EQUALS ZERO',
     $' ARE:')
7012 FORMAT(1X,I10)
7013 FORMAT ('1 CRITICAL COMPONENTS WHOSE VULNERABLE AREA EQUALS ZERO',
     $' ARE:'/' COMPONENT     PRESENTED AREA (SQ. FEET AND SQ. METERS)')
7014 FORMAT (' ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING')
      END
```

is used to define the input/output formats needed for program execution. This group of statements concludes Program PEAKAY.

SUBROUTINE PENT

This subroutine is called by Program PEAKAY for every shot line on the viewing plane and is used to compute the shot line penetration. Two nested DO loops are used to proceed along the shot line encounters at the time intervals specified by the user in the formatted PEAKAY input. The penetration times from the PEAKAY binary input file are used to compute component kill probabilities at each time interval. The component Pk's are used to compute system Pk's, total shot line Pk's, and cumulative Pk's for each component.

The statements

```
      SUBROUTINE PENT
      INTEGER    AVFLAG
      DIMENSION PKC(100),PKS(10)
      COMMON     R1(10,200),R3(200),I1(200),PKTIME(10,100),
     $           PKVAL(10,100),NOHITS(500),PLS(100),
     $           LOC(100),IST,NSHT,NENC,ICOMP(500),SHW(12,200),
     $           NOCOMP,COMPAV(500,10),GRID,TIMES(10),NTIME,IY(500),
     $           PMULT(10,10),IU(500),PAREA(500)
      COMMON    /ONE/ IFG(500),PHIT(4,500),AVFLAG
      LOGICAL    SNGSYS,MULSYS,IXSET
```

are used to declare array dimensions, provide for transfer of data into and out of this subroutine via common storage areas, and to declare the variable AVFLAG to be integer and the variables SNGSYS, MULSYS, and IXSET to be logical.

The statements

```
      PMX = 0.
      TBEG = 0.
      NEND = 1
      DO 10 II=1,10
        PKS(II) = 0.
   10 CONTINUE
```

are used to initialize the variables PMX, TBEG, NEND, and the PKS array.

The next statements

```
      DO 6000 NTIM=1,NTIME
        TEND = TIMES(NTIM)
        NBEG = NEND
```

are used to begin the time penetration loop. This DO loop is executed for every time interval specified in the user card input. Execution of the first assignment statement results in the assignation of the end time for this interval to the variable TEND. Execution of the other assignment statement results in the value of NBEG being set equal to the last encounter number that was not breached during a previous time interval.

The statements

```
      DO 1000 II=NBEG,NENC
        NEND = II
        PK = 0.
        PKC(II) = 0.
        SPK = 0.
        I = LOC(II)
        L = IY(I)
```

are used to start a DO loop which iterates for each encounter remaining on the shot line. The first assignment statement is used to transfer the encounter number for the iteration to the variable NEND. The next three assignment statements are used to assign initial values of 0.0 to the variables PK, PKC(II), and SPK. The last two assignment statements are used to assign the component number location to I and the system number to L.

The next two statements

```
SNGSYS = (L .NE. 0) .AND. (IU(I) .EQ. 0)
MULSYS = (L .NE. 0) .AND. (IU(I) .EQ. 1)
```

are used to assign values to two logical variables. The singly vulnerable system flag, SNGSYS, is assigned a value of TRUE only if the encountered component is in a system and its multiply vulnerable flag, IU, is 0. The multiply vulnerable system flag, MULSYS, is assigned a value of TRUE only if the encountered component is in a system and its multiply vulnerable flag is set to a value of 1.

The statements

```
           IF (TEND .LT. PKTIME(1,II)) GO TO 1900
           IF (PLS(II) .GE. PKTIME(1,II)) THEN
             IF (PKVAL(NOPNTS(I),I) .EQ. 1.0) THEN
               PK1 = AMIN1(PKTIME(NOPNTS(I),II),PLS(II))
             ELSE
               PK1 = PLS(II)
             END IF
             IF (TBEG .LT. PK1) THEN
               IX = 1
               IXSET = .FALSE.
 20            CONTINUE
                 IF ((IX .LT. NOPNTS(I)) .AND.
   $               (PKTIME(IX,II) .LT. TEND)) THEN
                   IX = IX + 1
                 ELSE
                   IXSET = .TRUE.
                 END IF
               IF (.NOT. IXSET) GO TO 20
               PK2 = PKTIME(IX-1,II)
               PK3 = AMIN1(TEND,PK1)
               IF ((PK3 .LE. PKTIME(NOPNTS(I),II)) .AND.
   $             (PK2 .NE. PKTIME(IX,II))) THEN
                 PK = (PK3 - PK2) / (PKTIME(IX,II) - PK2) *
   $               (PKVAL(IX,I) - PKVAL(IX-1,I)) + PKVAL(IX-1,I)
               ELSE
                 PK = PKVAL(IX,1)
               END IF
```

are used to compute the component probability of kill for one encounter. The first IF statement is used to branch out of the encounter loop to Statement 1900 if the end time for this interval is less than the minimum time required for a nonzero Pk. The first block IF ascertains whether the time for complete penetration is less than the minimum time for a nonzero Pk. If so, execution falls through to the end of the encounter loop. This event indicates that the component is completely perforated with a zero Pk, so the computation proceeds with the next encounter. The next block IF determines whether it is possible for the current component to be killed with a single shot line. If so, the maximum possible penetration time for this encounter, PK1, is the lesser of the times required to achieve a Pk of 1.0 and the time to achieve perforation of the component. If not, the maximum penetration time is the perforation time. The third block IF statement checks whether the maximum penetration time occurred during a previous time interval. If so, execution falls through to the end of the encounter loop. If not, the program initializes an index IX and a stopping condition flag IXSET for a "DO WHILE" type loop. The loop, beginning with Statement 20 and ending with the conditional GO TO, determines the minimum time associated with any of the entered Pk values that is greater than or equal to the current time interval. The contained block IF statement is used to increment the Pk time index until this value is achieved or until the last Pk time value is exceeded. The variable PK2, the minimum penetration time for this time interval, is set to the time associated with the last Pk value before the end of the time interval. PK3, the maximum penetration time for this time interval, is the lesser of the end of the time interval and the maximum possible penetration time. The final block IF statement is used to compute PK, the probability of kill for the encounter. If the maximum penetration time for the interval is less than or equal to the penetration time that causes the highest Pk and if the minimum penetration time is not equal to the time for the next Pk value. PK is interpolated linearly between the nearest input Pk values. Mathematically this is expressed as:

$$\boxed{Pk = \frac{(t_M - t_N)}{(t_{N+1} - t_N)} (Pk_{N+1} - Pk_N) + Pk_N} \tag{3-18}$$

If the conditions of the block IF are not met, the ELSE branch is executed, implying that either the component has been penetrated beyond the depth that causes its greatest Pk or that upon penetration to a certain depth, the Pk is discontinuous. In both cases, the Pk associated with the time found in the "DO WHILE" loop is the desired Pk, so PK is set to this value.

The statements

```
                  SPK = PK
                  IF (AVFLAG .EQ. 0) PK = PK * (1.0 - PMX)
                  PKC(II) = PK
                  IF (II .NE. 1) THEN
                    DO 172 LM = II-1,1,-1
                      IF (LOC(LM) .EQ. I) THEN
                        PK = PK * (1.0 - PKC(LM))
                        PKC(II) = PKC(LM) + PK
                        GO TO 175
                      END IF
172                 CONTINUE
                  END IF
175               CONTINUE
```

are used to compute the total shot line kill probability for the components by including the PK from any previous encounters. The first assignment statement is used to store the encounter PK value in the variable SPK, which is used for a system Pk computation. The IF statement is used to adjust the value of PK to obtain component incremental values if the area of vulnerability flag AVFLAG is reset; otherwise true values for component areas of vulnerability are obtained. The next statement is used to store the encounter Pk value in the PKC array. The succeeding block IF statement tests if this is the first shot line encounter. If so, program execution falls through to Statement 175. If this is not the first shot line encounter, the DO loop is used to search for the most recent shot line encounter for the same component. If no previous encounter involved the same component, the block IF statement falls through to the end of the DO loop on every iteration and the value of PKC(II) remains unchanged. If a previous encounter has the same component, the assignment statements are used to compute the probability of kill on the IIth encounter using:

$$\boxed{Pe_i = P_i \left(1 - \sum_{j=1}^{i-1} Pe_j\right)} \tag{3-25}$$

The probability of this exclusive event is then added to the previous shot line Pk and is stored in PKC(II). The GO TO statement is then used to branch out of the loop.

The next statement

```
      IF (SNGSYS) SPK = SPK * (1.0 - PKS(L))
```

is a logical IF statement used to adjust the system Pk value if the component is singly vulnerable. In this case, the system kill probability is the product of the encounter kill probability SPK, and the probability of system survival for all previous encounters (1-PKS(L)).

The statements

```
      IF (PKC(II) .GT. PRNT(3,I)) THEN
        PRNT(1,I) = SHW(1,IST)
        PRNT(2,I) = SHW(2,IST)
        PRNT(3,I) = PKC(II)
        PRNT(4,I) = PK3
      ELSE IF ((PKC(II) .EQ. PRNT(3,I)) .AND.
     $  (PRNT(4,I) .GT. PK3)) THEN
        PRNT(1,I) = SHW(1,IST)
        PRNT(2,I) = SHW(2,IST)
        PRNT(4,I) = PK3
      END IF
```

are used to store the maximum Pk encounter data for a component in the PRNT array. The block IF statement is used to compare the encounter Pk with the value already stored in the print array for the component. If the encounter Pk is greater than the value in the PRNT array, assignment statements store the shot line coordinates, the component kill probability, and the penetration time in the PRNT array. If the encounter Pk is not greater than the value in the PRNT array, the ELSE IF statement checks whether the encounter Pk is equal to the value in the PRNT array and whether the current penetration time is less than that stored in the PRNT array. If these conditions are met, the shot line coordinates and the penetration time are saved in the PRNT array. The END IF statement closes the block IF.

This statement

```
      IF (TEND .LT. PK1) GO TO 1900
```

is used to branch out of the encounter loop if the maximum Pk encounter time, PK1, exceeds the interval end time. If there is time remaining in the time interval after this encounter, the program continues with the next statements.

These statements

```
                    IF (SNGSYS) PKS(L) = PKS(L) + SPK
                    TBEG = PK3
                    DO  500 NTI = NTIM,NTIME
                      IF (SNGSYS) PMULT(L,NTI) = PMULT(L,NTI) + SPK
                      COMPAV(I,NTI) = COMPAV(I,NTI) + PK
      500           CONTINUE
```

are used to accumulate the system and component Pk's from this encounter, and to specify the start time. The first logical IF statement is used to sum the system Pk into the PKS array for singly vulnerable components only. The assignment statement is used to reassign the start time to equal the encounter end time. The DO loop is used to add the system Pk to the PMULT array, and the steps are inside of the encounter loop, the component Pk accumulation is expressed mathematically as:

$$\boxed{Pk_j = \sum_{i=1}^{N} Pe_i} \qquad (3\text{-}26)$$

The logical IF statement inside the loop is used to allow only singly vulnerable system kill probabilities to be added to the PMULT array.

The statements

```
                      IF ((.NOT. MULSYS) .AND. (AVFLAG .EQ. 1)) THEN
                        PMX = PMX + PK * (1.0 - PMX)
                      ELSE IF ((.NOT. MULSYS) .AND. (AVFLAG .EQ. 0)) THEN
                        PMX = PMX + PK
                      END IF
                    END IF
                  END IF
       1000     CONTINUE
```

are used to compute the shot line Pk and to end the encounter DO loop. The block IF is used to compute the shot line kill probability, excluding encounters with multiply vulnerable components. The shot line kill probability consists of the sum of the probabilities of exclusive events. If AVFLAG is set, the probability that the current encounter causes a kill is calculated using PK. If AVFLAG is reset, PK already equals the probability of kill for this encounter only. The three END IF statements end this block IF, the block IF where the beginning interval time is less than the maximum penetration time, and the block IF where the perforation time is at least equal to the time required for a nonzero Pk. Statement 1000 is used to end the encounter DO loop. This loop iterates until the time interval is completed or all components on the shot line are penetrated. Two IF statements inside the loop are used to branch to Statement 1900 when the time interval is completed. If all remaining components are penetrated during the time interval, the subroutine continues with the next statements.

The statements

```
          DO 1500 NTI = NTIM,NTIME
            SHW(NTI+2,IST) = PMX
 1500     CONTINUE
          RETURN
```

are used to store the shot line Pk for the remaining time intervals and to return control to Program PEAKAY. This is the end point of Subroutine PENT for a shot line which is able to penetrate all of the encountered components within the specified time intervals. The DO loop is used to store the shot line Pk values in array SHW beginning with the element for the current time interval NTIM and continuing through all remaining time intervals.

The next statements

```
 1900     CONTINUE
          IF (.NOT. MULSYS) THEN
            IF (AVFLAG .EQ. 1) THEN
              SHW(NTIM+2,IST) = PMX + PK * (1.0 - PMX)
            ELSE
              SHW(NTIM+2,IST) = PMX + PK
            END IF
          ELSE
            SHW(NTIM+2,IST) = PMX
          END IF
          IF (SNGSYS) PMULT(L,NTIM) = PMULT(L,NTIM) + SPK
          COMPAV(1,NTIM) = COMPAV(1,NTIM) + PK
 6000 CONTINUE
      RETURN
      END
```

are executed only after an encounter or series of encounters has used an entire time interval. This condition is detected by using two IF statements which invoke a branch to Statement 1900 from inside the encounter loop. The outer block IF is used to store the shot line Pk values after the NTIM time interval in the SHW array. The first block IF statement determines if the component is multiply vulnerable. If not, the inner block IF calculates the shot line kill probability, taking into account whether the probability of kill for one encounter PK is a true value or an incremental value (probability of kill on one encounter and survival on all others). If the component is multiply vulnerable, the ELSE branch of the outer block IF is executed and the shot line kill probability is unaffected by the current encounter. The branch to Statement 1900 is executed before updating the shot line kill probability PMX to include the last encounter. Hence, this step requires an assignment statement and an IF statement to store the Pk values from the last encounter. The logical IF statement is used to add the system Pk value to the PMULT array if the encountered component is singly vulnerable. The last assignment statement is used to accumulate the component Pk values in the COMPAV array. Statement 6000 is used to end the time penetration loop. If all time intervals are processed without complete shot line penetration, the RETURN statement is used to return control to Program PEAKAY.

Intentionally Left Blank

PROGRAM PRERD

This program should be used before Program QKPK. It is used to check for possible errors in the input cards. The output from this program includes a table of the component information arrays with flags marking possible errors and several summaries of the data arragned by various parameters. All output is on Logical Unit 6, the line printer. Table 4-2 lists the data checks which are made in this program, and Table 4-3 lists the summaries that are printed. Any flagged error should be checked and corrected by the user since this program does not correct errors for the user.

The first set of statements

```
 INTEGER    PKNBR
 COMMON     ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
$           TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
$           RHOF(500),IU(500),IY(500)
 COMMON     THINFL(500),NOCOMP
 COMMON    /LUNITS/ IRD,IWR
 COMMON    /SIZES/ ITC,IFX
 DIMENSION IE(34)
 DATA       IBLNK,ICHK,IASTER /' ',1H<,'*'/
 DATA       IRD,IWR /5,6/
 DATA       ITC,IFX,PKNBR /500,25,10/
```

is used to declare PKNBR as an integer, declare array dimensions, allocate common storage areas, and initialize the flag characters, input/output device numbers, and values of array dimensions. The initialization is done by using DATA statements.

The statements

```
OPEN (IRD,FILE='QKPKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
OPEN (IWR,FILE='PRERDOUT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
```

are used to connect logical units to the input and output files and to establish the connection properties between each unit/file pair.

The next two statements

```
 CALL READIN
 CALL FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,DEPTH,
$  PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
```

are used to execute two subroutines. Subroutine READIN is used to read the same input cards that Program QKPK will read and to make six of the tests listed in Table 4-2. Subroutine FSORT is used to sort the component information arrays into ascending order by component number.

The statements

```
WRITE (IWR,100)
WRITE (IWR,101)
```

are formatted WRITE statements which are used to print the heading for the component information table with flagged errors.

The next statements

TABLE 4-2. Data Tests Made by Program PRERD

Tested in Subroutine READIN:

1. Number of components exceeds component array dimensions (fatal error).
2. Number of points in the flux distribution exceeds array dimensions.
3. Flux level not within bounds (0.00 to 60000.0).
4. Number of points for which Pk values are to be entered exceeds array dimensions (fatal error).
5. Component information not in the expected format (fatal error).
6. Number of shot line time intervals for shot line breakdown not within array dimensions (fatal error).

Tested in Program PRERD:

1. Duplicate component identification numbers
2. Component identification number outside of the allowable range (1 to 9999)
3. Component criticality flag not equal to 0, 1, 2, or 3
4. Material identification code outside of the allowable range (1 to 11)
5. Table look-up code outside of the allowable range (1 to 9)
6. Material identification code and table look-up code both zero or both nonzero
7. Analysis type code not equal to 0, 1, or 2
8. Component initial operating temperature less than or equal to 0.0
9. Negative influence mode thickness, THINFL
10. Density factor RHOF outside of the allowable range (0.0 to 1.0)
11. Density factor and influence mode thickness both equal to 0.0 or both greater than 0.0
12. System number outside of the allowable range (0 to 10)
13. Singly vulnerable flag not equal to 0 or 1
14. System number equal to 0 and singly vulnerable flag not equal to 0
15. For depths of penetration and their associated probabilities P(K/H)
    a. Number of elements in DEPTH and PKVAL arrays outside of allowable range (2 to 10)
    b. Values of depth elements less than 0.0 or any two successive values decreasing in magnitude
    c. Values of probabilities of kill outside of the allowable range (0.0 to 1.0) and any two successive values decreasing

TABLE 4-3. Component Information Summaries From Program PRERD.

1. Component data with any of the possible errors from Table 4-2 indicated
2. Critical component summary and count
3. Component listing and count by material code
4. Component listing and count by table look-up code
5. Component listing and count by analysis type
6. Component listing and count by system number

```
          DO 110 J = 2,NOCOMP
            IF (ICOMP(J) .EQ. ICOMP(J-1)) THEN
              ICOMP(J) = -IABS(ICOMP(J))
              ICOMP(J-1) = -IABS(ICOMP(J-1))
            END IF
      110 CONTINUE
```

are used to test for duplicate component identification numbers. After executing Subroutine FSORT, the component numbers in array ICOMP are in increasing order, so this test can be done by looping through the array and comparing only the adjacent elements. This is done by using a DO loop and an IF statement. If duplicate component numbers are found, they are temporarily flagged by setting them to negative values.

The statements

```
          DO 199 J = 1,NOCOMP
            DO 120 I = 1,34
              IE(I) = IBLNK
      120   CONTINUE
```

are used to initiate a loop which will flag errors and print the component information arrays for every component. The inner DO loop in the above statements is used to assign blank characters to the IE array. If errors are found, this array is assigned a flag character in a position corresponding to the error type and printed with the component information arrays.

The statements

```
            IF (ICOMP(J) .LE. 0) IE(1) = ICHK
            ICOMP(J) = IABS(ICOMP(J))
            IF (ICOMP(J) .GT. 9999) IE(1) = ICHK
```

are used to finish the component identification number tests and set the first flag if an error is detected. The first IF statement is used to set the flag when the component number is negative or 0. The second IF statement is used to set the flag when the component number is greater than 9999. Any negative component numbers are made positive by using the IABS function. Since a negative number is also an indication of a duplicate component number, the first flag may indicate a duplicate component identification number or an identification number outside of the 1 to 9999 range.

The IF statement

```
            IF ((IFG(J) .LT. 0) .OR. (IFG(J) .GT. 3)) IE(2) = ICHK
```

is used to set the second flag when the criticality flag is not equal to 0, 1, 2, or 3.

The next statements

```
            IF ((ITAB(J) .EQ. 0) .AND. ((MAT(J) .LT. 1) .OR.
         $     (MAT(J) .GT. 11))) IE(3) = ICHK
            IF ((MAT(J) .EQ. 0) .AND. ((ITAB(J) .LT. 1) .OR.
         $     (ITAB(J) .GT. 9))) IE(4) = ICHK
            IF ((ITAB(J) .NE. 0) .AND. (MAT(J) .NE. 0)) THEN
              IE(3) = ICHK
              IE(4) = ICHK
            END IF
```

are used to test the table look-up code and the material code for the Jth component. The third flag is set if the table look-up code is 0 and the material identification code is less than 1 or greater than 11. The fourth flag is set if the material identification code is equal to 0 and the table look-up code is less than 1 or greater than 9. Both flags are set if both ITAB(J) and MAT(J) are non-zero.

The next IF statement

```
IF ((IANA(J) .LT. 0) .OR. (IANA(J) .GT. 2)) IE(5) = ICHK
```

is used to set the first flag if the analysis type code is not equal to 0, 1, or 2.

The statement

```
IF (TINIT(J) .LE. 0.) IE(6) = ICHK
```

is used to test the component's initial operating temperature and to set the sixth flag if it is less than or equal to 0.00.

The statements

```
      IF (THINFL(J) .LT. 0.) IE(9) = ICHK
      IF ((RHOF(J) .LT. 0.) .OR. (RHOF(J). GT. 1.)) IE(10) = ICHK
      IF (((THINFL(J) .GT. 0.) .AND. (RHOF(J) .GT. 0.)) .OR.
     $  ((THINFL(J) .LE. 0.) .AND. (RHOF(J) .LE. 0.))) THEN
        IE(9) = ICHK
        IE(10) = ICHK
      END IF
```

are used to test and possibly set the 9th and 10 flags. The first IF statement is used to set the 9th flag if the influence mode thickness, THINFL, is less than 0.00. The second IF statement is used to set the 10th flag when the density factor, RHOF, is less than 0.0 or greater than 1.00. Both of these flags are set if the influence thickness and the density factors are greater than 0.0, or if they are both less than or equal to 0.00.

The statements

```
IF ((IY(J) .LT. 0) .OR. (IY(J) .GT. 10)) IE(11) = ICHK
IF ((IU(J). LT. 0) .OR. (IU(J) .GT. 1)) IE(12) = ICHK
IF ((IY(J) .EQ. 0) .AND. (IU(J) .NE. 0)) THEN
  IE(11) = ICHK
  IE(12) = ICHK
END IF
```

are used to set the 11th and 12th flags if an error is detected in the system number array IY, or in the singly vulnerable flag array IU. The first IF statement is used to set the 11th flag when the system number is less than 0 or greater than 10. The next IF statement is used to set the 12th flag if the singly vulnerable flag is not equal to 0 or 1. Both flags are set if the system number is 0 and the singly vulnerable flag is not equal to 0.

The statements

```
            IF ((NOPNTS(J) .LT. 2) .AND. (NOPNTS(J) .GT. 10)) IF(13) = ICHK
            IF (DEPTH(1,J) .LT. 0.0) IE(14) = ICHK
            IF ((PKVAL(1,J) .LT. 0.0) .OR. (PKVAL(1,J) .GT. 1.0))
      $      IE(15) = ICHK
            IX = 2
  165       CONTINUE
              IF ((DEPTH(IX,J) .LT. 0) .OR. (DEPTH(IX,J) .LT. DEPTH(IX-1,J))
      $          IF(12+2*IX) = ICHK
              IF ((PKVAL(1,J) .LT. 0.0) .OR. (PKVAL(1,J) .GT. 1.0) .OR.
      $          (PKVAL(IX,J) .LT. PKVAL(IX-1,J))) IE(13+2*IX) = ICHK
              IX = IX + 1
            IF ((IX .LE. 10) .AND. (IX .LE. NOPNTS(J))) GO TO 165
```

are used to set the 13th through 33rd flags if an error is detected in the establishing of the laser penetration depths and the Pk's associated with them. The first IF statement is used to set the 13th flag when the number of points for penetration is less than 2 or greater than 10. The next IF statement is used to set the 14th flag when the minimum penetration for a non-zero Pk is negative. The third IF statement is used to set the 15th flag if the Pk associated with the first penetration depth is outside of the range 0.0 to 1.0. An index for the number of points entered is initialized at 2. A loop is then entered to check that each penetration depth is non-negative and that each successive depth is no less than the previous. If an error is detected, the next even flag is set. Likewise, the probability associated with each depth is checked to insure that it is within the range 0.0 to 1.0 and that it is no less than the previous Pk. If an error is detected, the next odd numbered flag is set. The index is incremented and the program branches to Statement 165 if NOPNTS penetration depths and Pk's have not been examined.

The next statements

```
            DO 170 I = 1,33
              IF (IE(I) .NE. IBLNK) THEN
                IE(34) = IASTER
                GO TO 180
              END IF
  170       CONTINUE
```

are used to test the flag array, IE, and to mark the line with an asterisk if it contains a possible error. If any of the flags from the previous tests have been set, the assignment statement is used to place an asterisk character in the 34th array position and the next GO TO statement is used to branch out of the DO loop to Statement 180. If none of the flags have been set, the DO loop executes for 33 iterations, always bypassing the assignment statement in the block IF.

The statements

```
  180     WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),IE(2),
      $     MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),IE(6),
      $     THINFL(J),IF(9),RHOF(J),IF(10),IY(J),IE(11),IH(J),IF(12),
      $     NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),PKVAL(K,J),
      $     IE(13+2*K),K = 1,NOPNTS(J))
  199 CONTINUE
```

are the last statements in the test loop, which is executed for every element in the component information arrays. The WRITE statement is used to print the component data for the Jth component with any flags that may have been set. After reaching Statement 199, the program returns to the beginning of the DO loop where the flag array, IE, is reinitialized and the tests will be executed again for the next component. This is repeated until all components have been tested.

The statements

```
      WRITE (IWR,200)
      WRITE (IWR,101)
      DO 210 I = 1,34
        IE(I) = IBLNK
  210 CONTINUE
```

begin the summary portion of this program. The WRITE statements are used to print the heading for the critical component summary. The DO loop is used to reset the IE array so that all characters are blank.

The next statements

```
      DO 235 I = 1,3
        NTYPE = 0
        DO 220 J = 1,NOCOMP
          IF (IFG(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $        IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $        IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $        IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $        PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
          END IF
  220   CONTINUE
        WRITE (IWR,230) NTYPE,I
  235 CONTINUE
```

are used to assemble and print the critical component summary. The criticality flag may have values of 0, 1, 2, or 3, but a 0 indicates a noncritical component, which will be excluded from this summary. The outer DO loop is used to test for the three remaining values. The variable NTYPE is used to count the number of each type of critical component, and is initialized to 0 at the start of this loop. The inner DO loop is used to cycle through the criticality flag array, IFG, for every component. Whenever the criticality flag matches I, the outer DO loop index, NTYPE, is incremented and the component information arrays are printed for the component. When the inner loop is completed, the last WRITE statement will be used to print the total number of components with the criticality flag I.

The statements

```
      WRITE (IWR,300)
      WRITE (IWR,101)
      DO 330 I = 1,11
        NTYPE = 0
        DO 310 J = 1,NOCOMP
          IF (MAT(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $        IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $        IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $        IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $        PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
          END IF
  310   CONTINUE
        WRITE (IWR,320) NTYPE,I
  330 CONTINUE
```

are used to assemble and write the component listing according to material types. The first two WRITE statements are used to print the heading for this summary. This is followed by two nested DO loops. The outer loop is used to initialize the type counter, NTYPE, and loop through the 11 possible material codes. The inner loop is used to test the material code for each component and if it is type I, the component information arrays are printed for the component. The final WRITE statement is used to print the total number of components, NTYPE, with a material type code I. Note that a component with a material code equal to 0 is excluded from this table.

The statements

```
      WRITE (IWR,400)
      WRITE (IWR,101)
      DO 430 I = 1,9
         NTYPE = 0
         DO 410 J = 1,NOCOMP
            IF (ITAB(J) .EQ. I) THEN
               NTYPE = NTYPE + 1
               WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $            IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $            IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $            IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $            PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
            END IF
  410    CONTINUE
         WRITE (IWR,420) NTYPE,I
  430 CONTINUE
```

are used to assemble and print a component summary by table look-up index. The first two WRITE statements are used to print the heading for this summary. This is followed by two nested DO loops which are used to test each component's table look-up index for each possible value and to print the component information for each component with a matching ıdex. This method was also used in assembling the two previous summaries.

The statements

```
      WRITE (IWR,500)
      WRITE (IWR,101)
      DO 530 I = 0,2
         NTYPE = 0
         DO 510 J = 1,NOCOMP
            IF (IANA(J) .EQ. I) THEN
               NTYPE = NTYPE + 1
               WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $            IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $            IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $            IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $            PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
            END IF
  510    CONTINUE
         WRITE (IWR,520) NTYPE,I
  530 CONTINUE
```

are used to assemble and print the component listing by analysis type. This procedure is the same as that used for the three preceding summaries. The analysis type summary groups the component information by analysis types 0, 1, and 2, by comparing analysis type values with I.

The following statements

```
      WRITE (IWR,600)
      WRITE (IWR,101)
```

```
          DO 630 I = 0,10
            NTYPE = 0
            DO 610 J = 1,NOCOMP
              IF (IY(J) .EQ. I) THEN
              NTYPE = NTYPE + 1
              WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $          IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $          IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $          IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $          PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
            END IF
      610   CONTINUE
          WRITE (IWR,620) NTYPE,I
      630 CONTINUE
```

are used to assemble and write the component listing by system number. The statements are used to execute the identical procedure used for the four previous summaries. These steps group and print the component information for system numbers 0 through 10.

The final group of statements

```
           STOP
      100 FORMAT ('1 THE FOLLOWING DATA HAS BEEN READ AND CHECKED FOR ',
         $    'ERRORS.'/'   AN ASTERISK IN THE LEFTMOST COLUMN INDICATES A',
         $    ' POSSIBLE ERROR IN THAT LINE.'/'   THE CHARACTER ',1H<,
         $    ' IS USED TO POINT TO THE POSSIBLY ERRONEOUS ITEM.')
      101 FORMAT (//
         1         '  ICOMP    IFG       MAT        ITAB       IANA     TINIT',
         2         5X,'THINFL     RHOF        IY         IU        NUPNTS',
         3         4X,'CARD'/
         4         9X,'DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)',
         5         2X,'PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)'/
         6         9X,'DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)'
         7         2X,'PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)'/)
      190 FORMAT (1X,A1,I5,A1,I6,A1,3(I9,A1),F11.2,A1,2(F9.2,A1),
         1         I7,A1,I9,A1,I10,A1,I9,2(/F16.2,A1,9(F9.2,A1)))
      200 FORMAT ('1 CRITICAL COMPONENT SUMMARY')
      230 FORMAT (/' THERE ARE ',I4,' CRITICAL COMPONENTS HAVING CRITICALI',
         $  'TY FLAG ',I3/)
      300 FORMAT ('1 COMPONENT LISTING BY MATERIAL TYPE')
      320 FORMAT (/' THERE ARE ',I4,' COMPONENTS OF MATERIAL TYPE ',I3/)
      400 FORMAT ('1 COMPONENT LISTING BY TABLE-LOOKUP INDEX')
      420 FORMAT (/' THERE ARE ',I4,' COMPONENTS HAVING TABLE INDEX ',I3/)
      500 FORMAT ('1 COMPONENT LISTING BY ANALYSIS TYPE')
      520 FORMAT (/' THERE ARE ',I4,' COMPONENTS OF ANALYSIS TYPE ',I3/)
      600 FORMAT ('1 COMPONENT LISTING BY SYSTEM NUMBER')
      620 FORMAT (/' THERE ARE ',I4,' COMPONENTS IN SYSTEM NUMBER ',I3/)
          END
```

are used to stop this program. The FORMAT statements are used to specify all output formats used by WRITE statements in this program.

SUBROUTINE PROPY

This subroutine is called by Subroutine RAT and is used to compute several properties for encountered metallic components. A large number of DATA statements are used to form a set of material property data from which the values are computed. The following properties are material dependent and are only computed once for each encounter: the coupling coefficient, component density, vapor temperature, melting temperature, and heat of fusion. The coupling coefficient is computed as the product of an intensity dependent factor and a thickness dependent factor. The interpolation scheme used to compute these two factors employs function values stored in arrays, but the argument values are generated in the coding. Two temperature dependent properties, thermal conductivity and specific heat, are computed every time this subroutine is executed. These two values are interpolated from arrays which contain both arguments and function values. All computed property values are transferred from this subroutine by using COMMON /PROP/.

The statements

```
      SUBROUTINE PROPY
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $            TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $            RHOF(500),IRVRS,IU(500),IY(500)
      COMMON    THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON   /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON   /PROP/ ALP,RHO,SVAL,TMLT,XLAMBD,RATE,JCOMP,N,DPM,CVAL,
     $            TVAP,CPL,ITER,T
      DIMENSION CONDUC(11,2,10),SPECIF(11,2,10),DENSIT(11),TMELT(11,2),
     $            HEATFU(11),ALPHA(11,16),ALPHAT(11,9)
```

are used to facilitate transferring values into and out of this subroutine and to specify array dimensions. The variables in the DIMENSION statement list are used to store the material property data for 11 material types.

The statements

```
C
C*****.....MATERIALS PROPERTY DATA.....*****
C
C ALL MATERIALS PROPERTY DATA IS VERY INITIAL.
C BARE MATERIALS-COUPLING COEFFFICIENT IS CONSIDERED TO BE A CONSTANT.
C COATED/PAINTED MATERIALS-COUPLING COEFFICIENT IS A FUNCTION OF INCIDENT
C FLUX AND MATERIALS THICKNESS.
C COUPLING COEFFICIENT DATA IS FOR NORMAL ANGLE OF INCIDENCE.
C ALPHA IS COUPLING COEFFICIENT AS A FUNCTION OF INCIDENT FLUX. ALPHAT
C IS THE CORRECTION FACTOR FOR MATERIAL THICKNESS.
C
C
C 2024 AL PAINTED SURFACE
C
      DATA CONDUC(1,1,1)/6./
      DATA (CONDUC(1,1,J),J=2,10)/76.,166.,279.,339.,429.,458.,3*0./
      DATA (CONDUC(1,2,J),J=2,10)/132.,156.,185.,189.,175.,169.,3*0./
      DATA SPECIF(1,1,1)/7./
      DATA (SPECIF(1,1,J),J=2,10)/ 20.,93.,205.,316.,400.,427.,450.,2*0./
      DATA (SPECIF(1,2,J),J=2,10)/849.,907.,966.,1025.,1096.,1129.,1154.,
     $      2*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
      DATA ALPHA(1,1)/.86/
      DATA (ALPHA(1,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $      .244,.224,.158,.136,.115,.1024,.094/
```

```
      DATA ALPHAT(1,1)/1.13/
      DATA (ALPHAT(1,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
      DATA DENSIT(1)/2770./
      DATA (TMELT(1,J),J=1,2)/502.,2500./
      DATA HEATFU(1)/389112./
C
C 7075 AL PAINTED SURFACE
C
      DATA CONDUC(2,1,1)/7./
      DATA (CONDUC(2,1,J),J=2,10)/3.,107.,207.,260.,336.,401.,428.,2*0./
      DATA (CONDUC(2,2,J),J=2,10)/122.,141.,177.,179.,177.,170.,167.,
     $      2*0./
      DATA SPECIF(2,1,1)/6./
      DATA (SPECIF(2,1,J),J=2,10)/0.,100.,200.,300.,400.,450.,3*0./
      DATA (SPECIF(2,2,J),J=2,10)/820.,903.,966.,1038.,1129.,1197.,3*0./
      DATA DENSIT(2)/2800./
      DATA HEATFU(2)/380744./
      DATA (TMELT(2,J),J=1,2)/476.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 2024 AL
C
      DATA ALPHA(2,1)/.86/
      DATA (ALPHA(2,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $      .244,.224,.158,.136,.115,.1024,.094/
      DATA ALPHAT(2,1)/1.13/
      DATA (ALPHAT(2,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C 5456 (AMG6) AL PAINTED SURFACE
C
      DATA CONDUC(3,1,1)/7./
      DATA (CONDUC(3,1,J),J=2,10)/0.,50.,100.,150.,200.,250.,300.,2*0./
      DATA (CONDUC(3,2,J),J=2,10)/109.,120.,128.,131.,132.,135.,136.,
     $      2*0./
      DATA SPECIF(3,1,1)/3./
      DATA (SPECIF(3,1,J),J=2,10)/100.,300.,650.,6*0./
      DATA (SPECIF(3,2,J),J=2,10)/920.,1046.,1255.,6*0./
      DATA DENSIT(3)/3000./
      DATA HEATFU(3)/389112./
      DATA (TMELT(3,J),J=1,2) /571.,1000./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 2024 AL
C
      DATA ALPHA(3,1)/.86/
      DATA (ALPHA(3,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $      .244,.224,.158,.136,.115,.1024,.094/
      DATA ALPHAT(3,1)/1.13/
      DATA (ALPHAT(3,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C 6AL4V TI PAINTED SURFACE
C
      DATA CONDUC(4,1,1)/7./
      DATA (CONDUC(4,1,J),J=2,10)/149.,260.,371.,427.,480.,538.,649.,
     $      2*0. /
      DATA (CONDUC(4,2,J),J=2,10)/9.5,11.8,14.,15.1,16.3,17.3,19.6,2*0./
      DATA SPECIF(4,1,1)/6./
      DATA (SPECIF(4,1,J),J=2,10)/94.,205.,316.,427.,538.,649.,3*0./
      DATA (SPECIF(4,2,J),J=2,10)/564.,594.,619.,640.,661.,678.,3*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
      DATA ALPHA(4,1)/.86/
      DATA (ALPHA(4,J),J=2,16)/.66,.51,.42,.35,.32,.29,.28,.26,.25,.24,
     $      .23,.23,.23,.23,.23/
      DATA ALPHAT(4,1)/1.59/
      DATA (ALPHAT(4,J),J=2,9)/1.04,.98,.98,.96,.96,.91,.87,.83/
      DATA (TMELT(4,J),J=1,2)/1565.,2277./
      DATA DENSIT(4)/4430./
      DATA HEATFU(4)/426870./
C
C PURE TI PAINTED SURFACE
C
      DATA CONDUC(5,1,1)/9./
      DATA (CONDUC(5,1,J),J=2,10)/ 27.,127.,227.,427.,627.,827.,1227.,
     $      1527.,1627./
      DATA (CONDUC(5,2,J),J=2,10)/21.9,20.4,19.7,19.4,20.2,21.3,24.5,
     $      27.1,28.0/
      DATA SPECIF(5,1,1)/7./
      DATA (SPECIF(5,1,J),J=2,10)/27.,127.,327.,627.,827.,927.,1127.,
     $       2*0./
      DATA (SPECIF(5,2,J),J=2,10)/422.,570.,677.,728.,743.,695.,695.,
     $       2*0./
      DATA (TMELT(5,J),J=1,2)/1660.,0./
      DATA HEATFU(5)/435136./
      DATA DENSIT(5)/4500./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 6AL4V TI
C
      DATA ALPHA(5,1)/.86/
      DATA (ALPHA(5,J),J=2,16)/.66,.51,.42,.35,.32,.29,.28,.26,.25,.24,
     $       .23,.23,.23,.23,.23/
      DATA ALPHAT (5,1)/1.59/
      DATA (ALPHAT(5,J),J=2,9)/1.04,.98,.98,.96,.96,.91,.87,.83/
C
C VM65-1 (ZK60) MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(6,1,1)/1./
      DATA (CONDUC(6,1,J),J=2,10)/25.,8*0./
      DATA (CONDUC(6,2,J),J=2,10)/109.,8*0./
      DATA SPECIF(6,1,1)/8./
      DATA (SPECIF(6,1,J),J=2,10)/152.,227.,327.,427.,520.,521.,627.,
     $       727.,0./
      DATA (SPECIF(6,2,J),J=2,10)/1063.,1075.,1205.,1031.,1380.,1305.,
     $       1364.,1599.,0./
      DATA HEATFU(6)/326352./
      DATA DENSIT(6)/1830./
      DATA (TMELT(6,J),J=1,2)/520.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(6,1)/.86/
      DATA (ALPHA(6,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $       .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(6,1)/1.13/
      DATA (ALPHAT(6,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C ML5 (AZ81A,AZ80) MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(7,1,1)/3./
      DATA (CONDUC(7,1,J),J=2,10)/0.,100.,203.,6*0./
      DATA (CONDUC(7,2,J),J=2,10)/64.9,73.6,80.6,6*0./
      DATA DENSIT(7)/1800./
      DATA HEATFU(7)/338904./
      DATA (TMELT(7,J),J=1,2)/490.,0./
      DATA SPECIF(7,1,1)/7./
      DATA (SPECIF(7,1,J),J=2,10)/127.,227.,327.,427.,507.,627.,727.,
     $       2*0./
      DATA (SPECIF(7,2,J),J=2,10)/1054.,1121.,1167.,1205.,1222.,1426.,
     $       1426.,2*0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(7,1)/.86/
      DATA (ALPHA(7,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $       .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(7,1)/1.13/
      DATA (ALPHAT(7,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C AZ31B MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(8,1,1)/3./
      DATA (CONDUC(8,1,J),J=2,10)/127.,234.,332.,6*0./
      DATA (CONDUC(8,2,J),J=2,10)/94.,101.,107.,6*0./
      DATA (TMELT(8,J),J=1,2)/605.,1107./
      DATA DENSIT(8)/1770./
      DATA HEATFU(8)/338904./
      DATA SPECIF(8,1,1)/8./
      DATA (SPECIF(8,1,J),J=2,10)/152.,227.,327.,527.,555.,556.,627.,
     $       727.,0./
      DATA (SPECIF(8,2,J),J=2,10)/1107.,1167.,1247.,1440.,1443.,1373.,
     $       1401.,1448.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(8,1)/.86/
      DATA (ALPHA(8,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $      .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(8,1)/1.13/
      DATA (ALPHAT(8,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C EI435 (IMONIC 75) NICKEL-CHROMIUM ALLOY PAINTED SURFACE
C
      DATA CONDUC(9,1,1)/7./
      DATA (CONDUC(9,1,J),J=2,10)/100.,200.,400.,600.,800.,1027.,1392.,
     $      2*0./
      DATA (CONDUC(9,2,J),J=2,10)/13.9,15.7,19.1,22.6,26.0,29.3,35.8,
     $      2*0./
      DATA HEATFU(9)/322168./
      DATA DENSIT(9)/8400./
      DATA (TMELT(9,J),J=1,2)/1400.,0./
      DATA SPECIF(9,1,1)/6./
      DATA (SPECIF(9,1,J),J=2,10)/100.,300.,500.,600.,750.,800.,3*0./
      DATA (SPECIF(9,2,J),J=2,10)/468.,502.,512.,623.,606.,619.,3*0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 304 STAINLESS STEEL
C
      DATA ALPHA(9,1)/.86/
      DATA (ALPHA(9,J),J=2,16)/.59,.43,.34,.30,.26,.22,.19,.18,.17,.16,
     $      .12,.12,.12,.12,.12/
      DATA ALPHAT(9,1)/1.91/
      DATA (ALPHAT(9,J),J=2,9)/1.04,.98,.96,.96,.46,.96,.96,.96/
C
C 304 (1KH18N9T) STAINLESS STEEL PAINTED SURFACE
C
      DATA DENSIT(10)/7990./
      DATA HEATFU(10)/297064./
      DATA (TMELT(10,J),J=1,2)/1400.,0./
      DATA CONDUC(10,1,1)/7./
      DATA (CONDUC(10,1,J),J=2,10)/200.,300.,400.,500.,650.,1027.,1392.,
     $      2*0./
      DATA (CONDUC(10,2,J),J=2,10)/18.,19.5,20.,21.3,25.,29.5,34.7,2*0./
      DATA SPECIF(10,1,1)/5./
      DATA (SPECIF(10,1,J),J=2,10)/200.,400.,600.,800.,1095.,4*0./
      DATA (SPECIF(10,2,J),J=2,10)/519.,553.,573.,598.,669.,4*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
      DATA ALPHA(10,1)/.86/
      DATA (ALPHA(10,J),J=2,16)/.59,.43,.34,.30,.26,.22,.19,.18,.17,.16,
     $      .12,.12,.12,.12,.12/
      DATA ALPHAT(10,1)/1.91/
      DATA (ALPHAT(10,J),J=2,9)/1.04,.98,.96,.96,.96,.96,.96,.96/
C
C CU BARE SURFACE
C
      DATA CONDUC(11,1,1)/5./
      DATA (CONDUC(11,1,J),J=2,10)/89.,232.,411.,449.,499.,4*0./
      DATA (CONDUC(11,2,J),J=2,10)/403.,394.,388.,387.,386.,4*0./
      DATA SPECIF(11,1,1)/9./
      DATA (SPECIF(11,1,J),J=2,10)/93.,204.,316.,427.,538.,760.,871.,
     $      982.,1065./
      DATA(SPECIF(11,2,J),J=2,10)/397.,402.,414.,423.,440.,473.,493.,
     $      513.,540./
      DATA DENSIT(11)/8960./
      DATA (TMELT(11,J),J=1,2)/1083.,2595./
      DATA HEATFU(11)/211710./
C
C  ALPHA IS A GUESS
C
      DATA (ALPHA(11,J),[illegible],16)/.4,15*0./
C
C
```

are DATA statements used to assign values to the material property data. Each of the 11 material types is identified in the comment statements preceding its group of DDATA statements. The contents of each array will be described along with the coding that uses the array.

The statements

```
MATL = MAT(N)
IF (ITER .EQ. 0) THEN
  T = TINIT(N)
  TVAP = TMELT(MATL,2)
  TMLT = TMELT(MATL,1)
  XLAMBD = HEATFU(MATL)
```

are used to obtain the material dependent characteristics from the materials property data. The first assignment statement is used to assign the material type for the component to the variable MATL. The last four assignment statements are used to determine the initial operating temperature T, the vapor temperature TVAP, the melting temperature TMLT, and the heat of fusion XLAMBD for the component in this encounter. The block IF statement is used to cause the execution of the statements in the THEN branch only when this is the first call to Subroutine PROPY for this encounter. The material characteristic variables are all in COMMON /PROP/ and, because of their global properties, retain their value on subsequent calls to this subroutine.

The statements

```
IF (MATL .GE. 11) THEN
  ALP = ALPHA(MATL,1)
```

are used to assign the coupling coefficient to the variable ALP if the encounter is with a bare metal. If the material code indicates a coated material, the ELSE branch of the block IF is executed.

The statements

```
          ELSE
            IF (PEAKF .GE. 1.0E+04) THEN
              XVAL = 1.0E+04
              K = 11
260           CONTINUE
                K = K + 1
                AVAL = K - 10
              IF (PEAKF .GT. XVAL*AVAL) GO TO 260
            ELSE
              XVAL = 1.0E+03
              K = 1
310           CONTINUE
                K = K + 1
                AVAL = K - 1
              IF (PEAKF .GE. XVAL*AVAL) GO TO 310
            END IF
            YVAL = ALPHA(MATL,K-1) - ALPHA(MATL,K)
            EM = -YVAL / XVAL
            ALPF = ALPHA(MATL,K) + EM * (PEAKF - XVAL * (AVAL))
```

are used to interpolate a value for the intensity dependent factor of the coupling coefficient for a coated material from the ALPHA array, using flux level arguments of 0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 20000, 30000, 40000, 50000, and 60000 watts/cm^2. The block IF statement determines whether the flux is at least 10000 watts/cm^2. The statements in this branch are used to find the interval number K that contains the flux level PEAKF. The arguments 10000, 20000, 30000, 40000, 50000, and 60000 are the product XVAL*AVAL in the IF statement and correspond to interval numbers 11, 12, 13, 14, 15, and 16. The ELSE branch of the block IF is executed only when the flux level PEAKF is less than 10000 watts/cm^2. The statements in this branch are used to find the interval K that contains the flux level PEAKF using flux arguments of 1000, 2000, 3000, . . . 10000, which correspond to interval numbers 2, 3, 4, . . . 11. The three assignment statements following the block IF are used to linearly interpolate a value for ALPF from the array ALPHA. This interpolation is written mathematically as:

$$\alpha_f = \alpha_K - \left(\frac{\alpha_{K-1} - \alpha_K}{\Delta X}\right)\left(f - X_K\right)$$

where

α_f	=	the interpolated result ALPF
α_K	=	the Kth function value from the ALPHA array
ΔX	=	the argument step size, XVAL
f	=	the flux level, PEAKF
X_K	=	the Kth argument value, (XVAL*AVAL)

The ALPHA array is doubly subscripted by the material type, MATL, and the interval number, K. Table 4-4 gives a tabular representation of the table being used for this interpolation scheme.

The next statements

```
DP = DPM / 0.0254
IF (DP .GE. 2.0E-01) THEN
  ALPT = ALPHAT(MATL,9)
```

are used to start the computation of the thickness dependent factor of the coupling coefficient. First, the component thickness is converted from meters to inches for use within Subroutine PROPY. The block IF statement is used to determine if the thickness DP is greater than or equal to the largest argument for the APHAT array. If it is, the THEN branch of the block IF is used to allocate the last element from array ALPHAT for the specific material type to the variable ALPT. If not, the ELSE branch is used to perform the interpolation.

The statements

```
         ELSE
           IF (DP .GE. 5.0E-02) THEN
             XVAL = 5.0E-02
             K = 6
360          CONTINUE
               K = K + 1
               AVAL = K - 5
             IF (DP .GE. XVAL*AVAL) GO TO 360
```

are used to determine the interval number, K, which contains the component thickness DP, if the thickness is between 0.05 and 0.20 inch. The block IF statement is used to determine if the thickness is at least 0.05 inch. If it is, the THEN branch is used to compare the arguments 0.10, 0.15, and

TABLE 4-4. ALPHA.

		MATERIAL CODES										
FLUX	K	1	2	3	4	5	6	7	8	9	10	11
0	1	.8600	.8600	.8600	.8600	.8600	.8600	.8600	.8600	.8600	.8600	.4000
1000	2	.7200	.7200	.7200	.6600	.6600	.7200	.7200	.7200	.5900	.5900	.0000
2000	3	.6000	.6000	.6000	.5100	.5100	.6000	.6000	.6000	.4300	.4300	.0000
3000	4	.4600	.4600	.4600	.4200	.4200	.4600	.4600	.4600	.3400	.3400	.0000
4000	5	.4000	.4000	.4000	.3500	.3500	.4000	.4000	.4000	.3000	.3000	.0000
5000	6	.3500	.3500	.3500	.3200	.3200	.3500	.3500	.3500	.2600	.2600	.0000
6000	7	.3200	.3200	.3200	.2900	.2900	.3200	.3200	.3200	.2200	.2200	.0000
7000	8	.2900	.2900	.2900	.2800	.2800	.2900	.2900	.2900	.1900	.1900	.0000
8000	9	.2700	.2700	.2700	.2600	.2600	.2700	.2700	.2700	.1800	.1800	.0000
9000	10	.2440	.2440	.2440	.2500	.2500	.2440	.2440	.2440	.1700	.1700	.0000
10000	11	.2240	.2240	.2240	.2400	.2400	.2240	.2240	.2240	.1600	.1600	.0000
20000	12	.1580	.1580	.1580	.2300	.2300	.1580	.1580	.1580	.1200	.1200	.0000
30000	13	.1360	.1360	.1360	.2300	.2300	.1360	.1360	.1360	.1200	.1200	.0000
40000	14	.1150	.1150	.1150	.2300	.2300	.1180	.1180	.1180	.1200	.1200	.0000
50000	15	.1024	.1024	.1024	.2300	.2300	.1072	.1072	.1072	.1200	.1200	.0000
60000	16	.0940	.0940	.0940	.2300	.2300	.1000	.1000	.1000	.1200	.1200	.0000

0.20, corresponding to interval numbers 7, 8, and 9 with the thickness DP. The argument values are the product XVAL*AVAL in the IF statement. The IF statement is used to loop back to Statement 360 until the interval K, which contains the thickness DP, is found.

The statements

```
                    ELSE
                      XVAL = 1.0E-02
                      K = 1
          410         CONTINUE
                        K = K + 1
                        AVAL = K - 1
                      IF (DP .GE. XVAL*AVAL) GO TO 410
                    END IF
```

are used to determine the interval K, which contains the thickness DP if the thickness is less than 0.05 inch. This search is similar to that executed by the previous group of statements, except that these steps use arguments of 0.01, 0.02, 0.03, 0.04, and 0.05 corresponding to interval numbers 2, 3, 4, 5, and 6, respectively.

The next statements

```
                  YVAL = ALPHAT(MATL,K-1) - ALPHAT(MATL,K)
                  EM = -YVAL/XVAL
                  ALPT = ALPHAT(MATL,K) + EM * (DP - XVAL * (AVAL))
                END IF
```

are used to linearly interpolate a value for the thickness dependent factor, ALPT, from the array ALPHAT, using the interval between subscripts K-1 and K, for the specified material code. Table 4-5 is a tabular representation of the function being used for this interpolation scheme. The interpolation scheme is written mathematically as:

$$\alpha_t = \alpha_K - \left(\frac{\alpha_{K-1} - \alpha_K}{\Delta X}\right)\left(d - X_K\right)$$

where

α_t = the interpolated result ALPT
α_K = the Kth function value from the ALPHAT array
ΔX = the argument step size, XVAL
d = the thickness DP
X_K = the Kth argument value, XVAL*AVAL

The statements

```
                  ALPC = ALPF * ALPT
                  IF ((ALPC .LT. ALPHA(MATL,1)) .AND. (ALPC .GT.
          $         ALPHA(MATL,16))) THEN
                    ALP = ALPC
                  ELSE IF (ALPC .GE. ALPHA(MATL,1)) THEN
                    ALP = ALPHA(MATL,1)
```

TABLE 4-5. ALPHAT.

		MATERIAL CODES										
THICKNESS	K	1	2	3	4	5	6	7	8	9	10	11
0.00	1	1.13	1.13	1.13	1.59	1.59	1.13	1.13	1.13	1.91	1.91	.00
0.01	2	1.07	1.07	1.07	1.04	1.04	1.07	1.07	1.07	1.04	1.04	.00
0.02	3	0.91	0.91	0.91	0.98	0.98	0.91	0.91	0.91	0.98	0.98	.00
0.03	4	0.84	0.84	0.84	0.98	0.98	0.84	0.84	0.84	0.96	0.96	.00
0.04	5	0.75	0.75	0.75	0.95	0.96	0.75	0.75	0.75	0.96	0.96	.00
0.05	6	0.68	0.68	0.68	0.96	0.96	0.68	0.68	0.68	0.96	0.96	.00
0.10	7	0.54	0.54	0.54	0.91	0.91	0.54	0.54	0.54	0.96	0.96	.00
0.15	8	0.38	0.38	0.38	0.87	0.87	0.38	0.38	0.38	0.96	0.96	.00
0.20	9	0.03	0.03	0.03	0.83	0.83	0.03	0.03	0.03	0.96	0.96	.00

```
        ELSE
          ALP = ALPHA(MATL,16)
        END IF
      END IF
```

are used to complete the computation of the coupling coefficient for coated materials. The assignment statement is used to compute this coefficient as the product of two factors: ALPF, which is a function of intensity and ALPT, which is a function of thickness. This is followed by a block IF which is used to compare the computed coupling coefficient with the endpoints on the curve represented by array ALPHA. If the computed coefficient is within the tabular values, it is assigned to the variable ALP. If it is outside the table range, the coupling coefficient ALP is assigned a value equal to the closer endpoint of the curve.

The statements

```
        RHO = DENSIT(MATL)
        TEMP = (TMLT + T) / 2.
        T = TEMP
      END IF
```

are used to compute the component density and temperature for the material type. The first assignment statement transfers the density for the component material code to the variable RHO. The last two assignment statements are used to set the variables TEMP and T equal to a temperature halfway between the initial operating temperature and the melt temperature. These are the last of the material dependent properties for this encounter. Since the variables ALP, RHO, and T are all global (in COMMON /PROP/), the next call to Subroutine PROPY for the same encounter can start immediately at the next set of statements.

The statement

```
      TEMP = T
```

begins the computation steps for the temperature dependent encounter properties. The temperature TEMP is the value which changes between calls to this subroutine and is the argument for the two interpolation procedures remaining in this subroutine.

The statements

```
      MVAL = NINT(CONDUC(MATL,1,1)) + 1
      IF (TEMP .LE. CONDUC(MATL,1,2)) THEN
        CVAL = CONDUC(MATL,2,2)
      ELSE IF (TEMP .GE. CONDUC(MATL,1,MVAL)) THEN
        CVAL = CONDUC(MATL,2,MVAL)
      ELSE
        DO 50 I = 2,MVAL
          IF (TEMP .LE. CONDUC(MATL,1,I)) THEN
            CVAL = CONDUC(MATL,2,I) - (CONDUC(MATL,1,I) - TEMP) *
     $          (CONDUC(MATL,2,I) - CONDUC(MATL,2,I-1)) /
     $          (CONDUC(MATL,1,I) - CONDUC(MATL,1,I-1))
            GO TO 100
          END IF
   50   CONTINUE
      END IF
```

are used to interpolate a value for CVAL, the thermal conductivity, corresponding to the temperature TEMP from the array CONDUC. The CONDUC array are used to interpolate a value for CVAL, the thermal conductivity, corresponding to the temperature TEMP from the array CONDUC. The CONDUC array has three subscripts: the first one identifies the material code, the second subscript equals 1 for the function arguments and equals 2 for the function values, and the third one corresponds to the points in the function table, beginning with the subscript equal to 2. The array element CONDUC(MATL,1,1) has the number of points in the function table, and CONDUC(MATL,2,1) is not used. The first assignment statement is used to store the number of data points in the integer variable NVAL. The second assignment statement is used to compute MVAL, the subscript of the last data point. The next three IF statements are used to determine if the temperature is within the function arguments. If the temperature is not within the function arguments, CVAL is assigned the function value corresponding to the argument closer to the temperature and the routine branches to Statement 100, bypassing the interpolation steps. The DO loop is used to find the argument interval containing the temperature TEMP. The IF statement in the loop is used to branch to the end of the loop until the correct interval is found. The assignment statement in the loop is used to perform a linear interpolation on the CONDUC array. The linear interpolation for CVAL is written mathematically as:

$$C_{val} = y_i - \left[\left(x_i - t\right)\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)\right]$$

where

y_i = the ith function value from the CONDUC array
x_i = the ith argument value from the CONDUC array
t = the temperature TEMP

The GO TO statement is used to branch out of the loop after a value for CVAL has been interpolated.

The next statements

```
      NVAL=SPECIF(MATL,1,1)+.001
      MVAL=NVAL+1
      SVAL=0.
      IF(TEMP.LE.SPECIF(MATL,1,2)) SVAL=SPECIF(MATL,2,2)
      IF(TEMP.GE.SPECIF(MATL,1,MVAL)) SVAL=SPECIF(MATL,2,MVAL)
      IF(SVAL.GT.0.) GO TO 200
      DO 150 I=2,MVAL
      IF(TEMP.GT.SPECIF(MATL,1,I)) GO TO 150
      SVAL=SPECIF(MATL,2,I)  -(SPECIF(MATL,1,I)-TEMP)*(SPECIF(MATL,2,I
     1)-SPECIF(MATL,2,I-1))/(SPECIF(MATL,1,I)-SPECIF(MATL,1,I-1))
      GO TO 200
  150 CONTINUE
  200 CONTINUE
      RETURN
      END
```

are used to interpolate a value for SVAL, the specific heat, from the array SPECIF. This array contains both arguments and function values in the same arrangement as the CONDUC array. The linear interpolation routine is identical

```
          ELSE
            ALP = ALPHA(MATL,16)
          END IF
        END IF
```

are used to complete the computation of the coupling coefficient for coated materials. The assignment statement is used to compute this coefficient as the product of two factors: ALPF, which is a function of intensity and ALPT, which is a function of thickness. This is followed by a block IF which is used to compare the computed coupling coefficient with the endpoints on the curve represented by array ALPHA. If the computed coefficient is within the tabular values, it is assigned to the variable ALP. If it is outside the table range, the coupling coefficient ALP is assigned a value equal to the closer endpoint of the curve.

The statements

```
          RHO = DENSIT(MATL)
          TEMP = (TMLT + T) / 2.
          T = TEMP
        END IF
```

are used to compute the component density and temperature for the material type. The first assignment statement transfers the density for the component material code to the variable RHO. The last two assignment statements are used to set the variables TEMP and T equal to a temperature halfway between the initial operating temperature and the melt temperature. These are the last of the material dependent properties for this encounter. Since the variables ALP, RHO, and T are all global (in COMMON /PROP/), the next call to Subroutine PROPY for the same encounter can start immediately at the next set of statements.

The statement

```
        TEMP = T
```

begins the computation steps for the temperature dependent encounter properties. The temperature TEMP is the value which changes between calls to this subroutine and is the argument for the two interpolation procedures remaining in this subroutine.

The statements

```
        MVAL = NINT(CONDUC(MATL,1,1)) + 1
        IF (TEMP .LE. CONDUC(MATL,1,2)) THEN
          CVAL = CONDUC(MATL,2,2)
        ELSE IF (TEMP .GE. CONDUC(MATL,1,MVAL)) THEN
          CVAL = CONDUC(MATL,2,MVAL)
        ELSE
          DO 50 I = 2,MVAL
            IF (TEMP .LE. CONDUC(MATL,1,I)) THEN
              CVAL = CONDUC(MATL,2,I) - (CONDUC(MATL,1,I) - TEMP) *
     $          (CONDUC(MATL,2,I) - CONDUC(MATL,2,I-1)) /
     $          (CONDUC(MATL,1,I) - CONDUC(MATL,1,I-1))
              GO TO 100
            END IF
   50     CONTINUE
        END IF
```

are used to interpolate a value for CVAL, the thermal conductivity, corresponding to the temperature TEMP from the array CONDUC. The CONDUC array has three subscripts: the first one identifies the material code, the second subscript equals 1 for the function arguments and equals 2 for the function values, and the third one corresponds to the points in the function table, beginning with the subscript equal to 2. The array element CONDUC(MAT,2,1) is not used. The first assignment statement is used to compute MVAL, the subscript of the last data point. The block IF is used to determine if the temperature is within the function arguments. If the temperature is not within the function arguments, CVAL is assigned the function value corresponding to the argument closer to the temperature. If the temperature is within the function arguments, the DO loop is used to find the argument interval containing the temperature TEMP. The IF statement in the loop is used to branch to the end of the loop until the correct interval is found. The assignment statement in the loop is used to perform a linear interpolation on the CONDUC array. The linear interpolation for CVAL is written mathematically as:

$$C_{val} = y_i - \left[\left(x_i - t\right)\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)\right]$$

where

y_i = the ith function value from the CONDUC array
x_i = the ith argument value from the CONDUC array
t = the temperature TEMP

The GO TO statement is used to branch out of the loop after a value for CVAL has been interpolated.

The next statements

```
100 MVAL = NINT(SPECIF(MATL,1,1)) + 1
    IF (TEMP .LE. SPECIF(MATL,1,2)) THEN
      SVAL = SPECIF(MATL,2,2)
    ELSE IF (TEMP .GE. SPECIF(MATL,1,MVAL)) THEN
      SVAL = SPECIF(MATL,2,MVAL)
    ELSE
      DO 150 I = 2,MVAL
        IF (TEMP .LE. SPECIF(MATL,1,I)) THEN
          SVAL = SPECIF(MATL,2,I) - (SPECIF(MATL,1,I) - TEMP) *
   $        (SPECIF(MATL,2,I) - SPECIF(MATL,2,I-1)) /
   $        (SPECIF(MATL,1,I) - SPECIF(MATL,1,I-1))
          GO TO 200
        END IF
150   CONTINUE
    END IF
200 CONTINUE
    RETURN
    END
```

are used to interpolate a value for SVAL, the specific heat, from the array SPECIF. This array contains both arguments and function values in the same arrangement as the CONDUC array. The linear interpolation routine is identical to the one used in the preceding group of statements. Interpolation for SVAL is written mathematically as:

$$S_{val} = y_i - \left[\left(x_i - t\right)\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)\right]$$

where

y_i = the ith function value from the SPECIF array
x_i = the ith argument value form the SPECIF array
t = the temperature TEMP

The RETURN and END statements are used to return control to Subroutine RAT and end this program unit.

PROGRAM QKPK

This program requires two input files: a formatted file read by Subroutine RDATA and a binary LOS file for one view, which is the output of either Program CONMAG, FASTGEN, or SHOTGEN. These data are used to compute from three to eleven penetration times for each critical encounter along each shot line: the minimum time needed to achieve a nonzero kill probability, the minimum times needed to achieve from zero to eight intermediate kill probabilities, the minimum time needed to achieve a maximum kill probability (less than or equal to 1.0), and the time needed to completely perforate the component. This information is written on a binary output file which is used by Program PEAKAY to compute component kill probabilities and vulnerable areas. Program QKPK is also used to compute and print a breakdown of critical shot lines by times needed to reach maximum Pk. A critical shot line has at least one encounter with a critical component. A shot line reverse flag is provided so that a user may obtain data for the opposite view without generating a new LOS tape.

The first set of statements

```
      INTEGER    PKNBR
      COMMON     ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $           TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $           RHOF(500),IRVPS,IU(500),IY(500)
      COMMON     THINEL(500),SH(2,170),JH(5,170),CINCH
      COMMON    /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON    /TWO/ XLOS(100),JHT(100),NFNC,OHD(100)
      COMMON    /THREE/ PKTIME(10,200),PLS(200),T1(200)
      COMMON    /FOUR/ RATES(100,25),FTIM(25),FXCM(25),FXM(25),IFMAX,
     $           TIMAX(25),MAXT(27),NIMAX
      COMMON    /UNITS/ IRD,IWR,IIN,IOUT
      COMMON    /SIZES/ ITC,IFX,IFXX
      DIMENSION THRAT1(500),THRAT2(500)
```

is used to declare the variable PKNBR as an integer, to allocate common storage area for data transfer among subroutines, and to declare array dimensions.

The statements

```
      DATA       ITC,MAXENC,PKNBR,IFX,IFXX /500,100,10,25,27/
      DATA       IRD,IWR,IIN,IOUT /5,6,1,2/
      DATA       MAXT /27*0/
      DATA       MAT(499)/1/,MAT(500)/2/,IFG(499)/0/,IFG(500)/0/
      DATA       ITAB(499)/0/,ITAB(500)/0/,IANA(499)/2/,IANA(500)/1/
      DATA       TINIT(499)/110./,TINIT(500)/35./,IY(499)/0/,IY(500)/0/
      DATA       RHOF(499),RHOF(500)/1.,1./
      DATA       IECHO /1/
```

are DATA statements which are used to initialize the variables whose names appear in the DATA statement lists. The first DATA statement is used to initialize variables with values of current array dimensions. The second DATA statement is used to initialize the input and output device numbers. The next DATA statement is used to zero the MAXT array. The DATA statements for the 499th and 500th positions in the component information arrays are used to assign default component characteristics. These data will be used if the shot line data (binary file) describe an encounter with a component which has no component information input (formatted file). The final DATA statement is used to set the echo flag so that all binary output will also be written at the line printer.

The statements

```
OPEN (IRD,FILE='QKPKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
OPEN (IWR,FILE='QKPKPRINT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
OPEN (IIN,FILE='QKPKTAPEIN',FORM='UNFORMATTED',RECFM='VARIABLE')
OPEN (IOUT,FILE='QKPKTAPEOUT',FORM='UNFORMATTED',RECFM='VARIABLE')
```

are used to connect logical units to the input and output files and to establish the connection properties between each unit/file pair.

The statements

```
      CALL RDATA
      IF (NTMAX .LE. 0) THEN
        NTMAX = IFMAX
        DO 3 I = 1,NTMAX
          TIMAX(I) = FTIM(I)
    3   CONTINUE
      END IF
```

are used to call Subroutine RDATA to read all formatted input for this program, and to fill the TIMAX array. The time intervals for shot line breakdown, TIMAX, is an optional input. The IF statement is used to determine if this input was included. If the values are not included, the DO loop and assignment statements are executed. The result is that array TIMAX contains values equal to the time intervals from the flux distribution and NTMAX equals the number of time intervals.

The statements

```
READ (IIN) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,RADEXP
IF (CINCH .LE. 0.) CINCH = 1.0
CMETER = CINCH*0.0254
GRID = GRID * CINCH
YMAX = YMAX * CINCH
YMIN = YMIN * CINCH
ZMAX = ZMAX * CINCH
ZMIN = ZMIN * CINCH
```

are used to read the first record from the binary file and to convert the minimum and maximum viewing plane coordinates and grid sizes to inches. The conversion factor CINCH is assigned a default value of 1.0 by using the first IF statement whenever it has been read with a value less than or equal to 0.0. The factor for conversion from input units to meters, CMETER, is also computed.

The next statements

```
IF (IRVRS .EQ. 1) THEN
  Y1  = -YMAX
  YMAX = -YMIN
  YMIN = Y1
  EL = -EL
  AZ = AZ - 180.
  IF (AZ .LT. 0.) AZ = AZ + 360.
END IF
```

are used to reverse the viewing plane data whenever the shot line reverse flag is set. The first IF statement is used to test IRVRS, the shot line reverse flag. If it is equal to 1, the flag is set and the assignment statements are used to interchange and negate the minimum and maximum y-coordinates and to compute the reversed view attack angles. The last IF statement is used to maintain a positive value for the azimuth attack angle.

The statement

```
      CALL FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,DEPTH,
     $  PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
```

is a CALL statement used to invoke Subroutine FSORT which sorts the component information arrays into ascending order by component number. The component numbers are the values in array ICOMP. The sort subroutine maintains correspondence among all the array arguments by subscript.

The statements

```
      WRITE (IWR,1003)
      DO 50 J = 1,NOCOMP
        WRITE (IWR,1004) ICOMP(J),IFG(J),MAT(J),ITAB(J),IANA(J),
     $    TINIT(J),THINFL(J),RHOF(J),IY(J),IU(J),J,(DEPTH(I,J),
     $    PKVAL(I,J),I=1,NOPNTS(J))
        DO 40 I = 1,NOPNTS(J)
          DEPTH(I,J) = DEPTH(I,J) * CMETER
   40   CONTINUE
        THINFL(J) = THINFL(J) * CMETER
   50 CONTINUE
```

are used to print the component information arrays on the line printer and to convert DEPTH and THINFL values to meters. The first write statement is used to print the heading. The outer DO loop is used to print the component information and to convert the influence mode wall thickness THINFL to meters. The inner DO loop is used to convert each laser penetration depth DEPTH to meters.

The next statements

```
      NENC = 0
      IFILL = 0
      IFM = IFMAX - 1
      IFXM = IFXX - 1
      WRITE (IOUT) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
      WRITE (IOUT) (ICOMP(I),IFG(I),IY(I),IU(I),NOPNTS(I),(PKVAL(J,I),
     $  J=1,NOPNTS(I)),I=1,NOCOMP),IRVRS,IFMAX,(FTIM(I),FXCM(I),I=1,
     $  IFMAX)
       IF (IECHO .EQ. 1) THEN
          WRITE (IWR,1000) IOUT
          WRITE (IWR,1014) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
          WRITE (IWR,1015)
          DO  60 I = 1,NOCOMP
            WRITE (IWR,1016) ICOMP(I),IFG(I),IY(I),IU(I),
     $        NOPNTS(I),(PKVAL(J,I),J=1,NOPNTS(I))
   60     CONTINUE
          WRITE (IWR,1017) IRVRS,IFMAX,(FTIM(I),FXCM(I),I = 1,IFMAX)
       END IF
```

are used to initialize four variables, to write the first two records of the binary output file, and if the echo flag is set, to print the binary output values on the line printer. The variable NENC is used to count the number of encounters for each shot line and the variable IFILL to count the number of elements in the binary output arrays. The first two assignment statements are used to set these two counters equal to 0. The variable IFM is used to reference the next to last element in the flux arrays and variable IFXM is used to reference the next to last element in the array MAXT, which counts the critical shot line breakdown. The first binary WRITE statement is used to write the viewing plane description on the first record of the binary output file. The second WRITE statement to device IOUT is used to write the component number array, criticality flag array, system number array, multiple vulnerability flag array, reverse flag, number of points in the flux distribution, and flux distribution with time intervals. The block IF is used to print the binary output values on the line printer when the echo flag is set.

The statements

```
100 CONTINUE
    READ (IIN) (DUM,(SH(I,J),I=1,2),(JH(I,J),I=1,5),J=1,170)
      DO 110 J = 1,170
```

are used to read the next record of LOS data and to initiate a DO loop to process these data. The LOS data on the binary input file are arranged in groups of 170, and are stored in 170 consecutive array locations by using an implied DO loop. Consequently, a DO loop must be executed 170 times to process each set of LOS data. When the DO loop execution has been completed, a GO TO statement is used to return to Statement 100 so that the next set of LOS data may be read and processed.

The statements

```
IF (JH(2,J) .EQ. 0) GO TO 5000
ICODE = MOD(JH(1,J),10)
NENC = NENC + 1
IF (NENC .GT. 100) GO TO 530
```

are used to test for the end of view indicator, unpack the end of shot line indicator, increment the number of encounters, and test for a bounds violation. The first IF statement is used to test for an end of view, JH(2,J) = 0, and if true, causes a branch to Statement 5000. This is the normal exit from the LOS data processing loop. The assignment statement for the variable ICODE uses the FORTRAN MOD function to unpack the right-most digit of JH(1,J). The last two executable statements increment NENC, the number of encounters per shot line, and branch to Statement 530, where a fatal error message is printed if NENC is greater than 100. If NENC is small enough, the program continues with the next set of statements.

These statements

```
XLOS(NENC) = FLOAT(JH(3,J)) * 0.01 * CMETER
JHT(NENC) = JH(2,J)
```

are used to compute the line of sight thickness in meters from JH(3,J) and store it in array XLOS. The last assignment statement is used to store the component number for encounter NENC in array JHT.

The statements

```
IF (IRVRS .EQ. 0) THEN
  OBQ(NENC) = FLOAT(JH(4,J)) * 0.001
ELSE
  OBQ(NENC) = FLOAT(JH(5,J)) * 0.001
END IF
```

are used to store the secant of the entrance obliquity angle in array OBQ. The IF statement is used to test the shot line reverse flag. If the shot line is not reversed, (IRVSR = 0), the program uses JH(4,J) for the secant of the obliquity angle; otherwise the program uses JH(5,J). Since the binary LOS file actually stores 1000 times the secant value, the JH array element value is multiplied by 0.001 to obtain the true secant value.

The statement

```
                  IF (ICODE .EQ. 9) THEN
```

is used to test for the end of shot line flag. If ICODE does not equal 9, the end of the shot line has not been reached and the program bypasses the block IF and continues execution at Statement 110, the last statement in the shot line data processing loop. Then, the program will branch to process more shot line data. When the end of the shot line is reached, the next statements will be executed.

These statements

```
              IF (IRVRS .EQ. 1) CALL REVRSE(J)
              DO 170 IR = 1,IFMAX
                PEAKF = FXCM(IR)
                FLX = FXM(IR)
                CALL RAT (RATES(1,IR),IR,MAXENC)
                IF (IFLAG .EQ. 0) GO TO 180
170           CONTINUE
```

are used to reverse the shot line, if required, and to compute the penetration rate for each flux level. The IF statement is used to test the shot line reverse flag, and if it is set, to call Subroutine REVRSE, which is used to interchange the order of the components along the shot line. The DO loop is used to compute the penetration rate at each level of flux. Assignment statements for variables PEAKF and FLX are used to store the flux level in COMMON /ONE/. These values are accessed by either of the subroutines PROPY or TABLE, depending upon which is called by the Subroutine RAT. The CALL statement is used to invoke Subroutine RAT which computes the rate of penetration at the IRth flux level for each component along the shot line. The argument array, RATES, is used to store the penetration rates. The first subscript for RATES is the encounter number and the second subscript is the flux index. The variable IFLAG is computed in Subroutine RAT and equals the number of the last critical encounter. If IFLAG equals 0, there are no critical components on the shot line and the IF statement is used to branch out of the loop.

The statements

```
180           IFILL = IFILL+1
              PKTIME(1,IFILL) = SH(1,J) * CINCH
              PKTIME(2,IFILL) = SH(2,J) * CINCH
              DO 190 K = 3,PKNHR
                PKTIME(K,IFILL) = 0.0
190           CONTINUE
              PLS(IFILL) = 0.
              II(IFILL) = NCWTT
```

are used to store the next group of data in the binary output arrays. The first assignment statement is used to increment IFILL, the subscript for the binary output arrays. The shot line y- and z-coordinates are converted to inches and stored in the first two elements of each row of the PKTIME array. The remaining elements in each row are set to 0.0 in the DO loop. A 0.0 is assigned to array PLS and the number of critical encounters on the shot line is stored in array I1 by executing the last two assignment statements.

The statements

```
              IF (NCRIT .EQ. 0) MAXT(IFXX) = MAXT(IFXX) + 1
              IF (IFILL .EQ. 200) THEN
                WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),I1(JX),
     $            JX=1,200)
                IF (IECHO .EQ. 1)
     $            WRITE (INW,1002) ((PKTIME(IX,JX),IX=1,10),PLS(JX),
     $            I1(JX),JX=1,200)
                IFILL = 0
              END IF
              IF (IFLAG .NE. 0) THEN
```

are used to count the number of noncritical shot lines, write the binary output arrays (if they are full), and fall through to the end of the LOS data processing loop (if the shot line has no critical encounters). The first IF statement is used to add 1 to the last element in array MAXT, the number of noncritical shot lines if NCRIT, the number of critical encounters, equals 0. The next IF statement is used to determine if the binary output arrays are full. When IFILL equals 200, the arrays are written to device IOUT, using the binary WRITE statement and echoed to the line printer if the echo flag is set. The subscript IFILL is reset to 0 after the WRITE statements are executed. The binary output steps are skipped if IFILL is less than 200. The last IF statement is used to determine if there are no critical encounters and to bypass the block IF and continue execution at Statement 110.

The next statements

```
                TUSED = 0.
                TMAX = 0.
                TMAX1 = 1.E30
                IRUSED = 1
  401           CONTINUE
                DO 499 II = 1,NENC
```

are only executed if the shot line has at least one critical encounter. They are used to initialize four variables and to initiate a DO loop which iterates for every encounter on the shot line.

The statements

```
                  RBAR = 0.
                  IRNOW = IRUSED
                  TNOW = TUSED
                  DISLFT = XLOS(II)
```

are used to initialize four variables and are the first steps in the encounter loop. The variable RBAR is the penetration rate and IRNOW is a subscript for the flux distribution array. TNOW represents the time used up to the present step. DISLFT is the distance remaining to be penetrated in encounter II.

The statements

```
410         CONTINUE
              RATE = RATES(II,IRNOW)
              IF (RATE .LE. 0.) GO TO 430
              IF (DISLFT .LE. (FTIM(IRNOW) - TNOW) * RATE) GO TO 420
              DISLFT = DISLFT - (FTIM(IRNOW) - TNOW)*RATE
              TNOW = FTIM(IRNOW)
              IRNOW = IRNOW + 1
            IF (IRNOW .LE. IFMAX) GO TO 410
            IRNOW = IRNOW - 1
```

are used to compute the time step during which penetration ends for the IIth encounter on the shot line. The first assignment statement is used to set the variable RATE equal to the penetration rate for the IIth encounter and the IRNOWth time step in the flux distribution. The first IF statement is used to branch to Statement 430 when the rate is 0.00. The second IF statement is used to branch to Statement 420 if the penetration is completed during time step IRNOW. If both tests fail, two assignment statements are used to compute DISLFT, the distance left to penetrate after time step IRNOW, and TNOW, the time after this time step. Next, IRNOW is incremented and the IF statement is used to branch back to Statement 410 to compute penetration for the next step in the flux distribution. If these steps iterate through the entire flux distribution without complete penetration, the last IF statement fails, and IRNOW is decremented so that it points to the last step in the distribution.

The statements

```
420         CONTINUE
            TNOW = TNOW + DISLFT / RATE
            TLOS = TNOW - TUSED
            RBAR = XLOS(II) / TLOS
```

are used to compute TNOW, the time of complete component penetration, and RBAR, the average penetration rate for the encounter. These steps are bypassed if the penetration rate becomes 0.0 before complete penetration.

The statements

```
430         CONTINUE
            J = JMT(II)
            IF (IFG(I) .NE. 0) THEN
              IFILL = IFILL + 1
              PLS(IFILL) = TNOW
              I1(IFILL) = JMT(II)
              THRAT1(I) = AMAX1(THRAT1(I),DEPTH(1,I) * OBQ(II) /
     $          XLOS(II))
              THRAT2(I) = AMAX1(THRAT2(I),DEPTH(NOPNTS(I),I) *
     $          OBQ(II) / XLOS(II))
```

are used to store the data for complete penetration in the binary output arrays. If the encountered component is a critical one, the statements in the block IF are executed; otherwise the program continues execution at Statement 110, the last statement in the shot line data processing loop. Three assignment statements are used to increment IFILL, to store the time for complete penetration in the PLS array, and to store the component number location in array I1. The values for insertion in the THRAT1 and THRAT2 arrays are computed using the AMAX1 function and are used to store the ratios DEPTH(1)*OBQ/XLOS and DEPTH(NOPNTS)*OBQ/XLOS for the encounter with the smallest non-zero line of sight thickness for the Ith component.

The statements

```
                      DO 440 K = 1,10
                        PKTIME(K,IFILL) = 0.0
  440                 CONTINUE
                      IF (RBAR .GT. 0.) THEN
                        DO 450 K = 1,NOPNTS(I)
                          PKTIME(K,IFILL) = DEPTH(K,I) * OBQ(II) / RBAR +
     $                      TUSED)
  450                   CONTINUE
```

are used to compute and store the time required to penetrate to the specified depth to obtain the associated Pk. The first DO loop is used to initialize the PKTIME array to zero. The IF block is executed if the average penetration rate for the component is greater than zero. The DO loop in the block IF is used to compute the total time required to penetrate the component to obtain the Pk values entered in the array PKVAL.

The statement

```
                      IF ((PKTIME(NOPNTS(I),IFILL) .LE. PLS(IFILL))
     $                  .AND. (PKVAL(NOPNTS(I),I) .EQ. 1.0))
     $                  TMAX1 = AMIN1(TMAX1,PKTIME(NOPNTS(I),IFILL))
```

is used to compute TMAX1, the earliest time that a component Pk equal to 1.0 occurs on this shot line. The IF statement is used to exclude encounters which complete penetration before attaining Pk = 1.00.

The next statements

```
                      IF (PKVAL(NOPNTS(I),I) .EQ. 1.0) THEN
                        TMAX = AMAX1(TMAX,AMIN1(TNOW,PKTIME(NOPNTS(I),
     $                    IFILL)))
                      ELSE
                        TMAX = AMAX1(TMAX,AMIN1(TNOW,PLS(IFILL)))
                      END IF
```

are used to compute TMAX, the minimum time needed to process all components on this shot line. The block IF statement determines whether it is possible to guarantee a kill of a component with a single shot line. If so, the assignment statement in the THEN branch is executed; otherwise, the ELSE branch is used. The AMIN1 function is used to select the minimum time to complete the current encounter. The AMAX1 function is used to select the greatest time for all encounters by comparing the value of TMAX from all previous encounters with the time for the current encounter. A completed encounter requires complete penetration of the component or a Pk equal to 1.0, whichever occurs first.

The statements

```
                      IF (IFILL .EQ. 200) THEN
                        WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),
     $                    II(JX),JX=1,200)
                        IF (IFCHO .EQ. 1)
     $                    WRITE (IWR,1002) ((PKTIME(IX,JX),IX=1,10),
     $                      PLS(JX),II(JX),JX=1,200)
                        IFILL=0
                      END IF
```

```
                END IF
              END IF
              TUSED = TNOW
              IRUSED = IRNOW
  499       CONTINUE
```

are used to write another record of the binary output file, if the binary output arrays are full, and to end the encounter DO loop. The block IF statement is used to determine when the binary output arrays are full. When they are full, the binary WRITE statement is executed and if the echo flag is set, the formatted WRITE statement is executed. The output array index, IFILL, is reset equal to 0 following the WRITE statement execution. The block IF statements determining whether the binary output arrays are full, whether the average penetration rate is greater than zero, and whether the encountered component is a critical one are closed with the three END IF's. The next assignment statement is used to set TUSED equal to the time for complete penetration of this encounter and the last assignment statement is used to set IRUSED equal to the flux step number during which complete penetration was attained. Statement 499 is used to end the encounter DO loop.

The statements

```
            TMAX1 = AMIN1(TMAX,TMAX1)
            DO 500 I=1,NTMAX
              IF (TMAX1 .LE. TIMAX(I)) THEN
                MAXT(I) = MAXT(I) + 1
                GO TO 510
              END IF
  500       CONTINUE
            MAXT(IFXM) = MAXT(IFXM) + 1
  510       CONTINUE
```

are executed after each shot line is processed and are used to compute the breakdown of the shot line by the time required to achieve maximum Pk. First, TMAX1 is recomputed to be the minimum of the time needed to completely process the shot line and the time needed to reach a component Pk = 1.00. The DO loop is then used to cycle through the time intervals for shot line breakdown. The IF statement is used to find the time interval I in which TMAX1 falls. After the time bin I is found, the assignment statement is used to increment MAXT(I) and the GO TO statement is executed to branch out of the loop. The assignment statement following Statement 500 is executed only if TMAX1 is greater than all of the time intervals for shot line breakdown. It is used to increment the next to last array location of MAXT.

The statements

```
              NENC=0
            END IF
          END IF
  110   CONTINUE
      GO TO 100
```

represent the end of the LOS data processing loop. The assignment statement is used to reset the number of shot line encounters to 0 and is executed after a complete shot line has been processed. The END IF statements close the block IF determining whether the shot line has any critical encounters and whether the end of the shot line has been reached. Statement 110 is used to end the DO loop, which is executed for each of the 170 LOS data array elements

The GO TO statement is used to branch back to read and process a new set of LOS data.

The next statements

```
5000 CONTINUE
     IFILL = IFILL + 1
     DO 120 J = IFILL,200
       I1(J) = 9999
 120 CONTINUE
```

are executed after an end of view has been detected in the binary LOS input data. The first assignment statement is used to increment IFILL, the number of values in the binary output arrays. The DO loop then is used to fill the remainder of the binary output array, I1, with an end of view flag, 9999.

The statements

```
      WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),I1(JX),JX=1,200)
      IF (IECHO .EQ. 1) THEN
         WRITE (IWR,1002) ((PKTIME(IX,JX),IX=1,10),
     $       PLS(JX),I1(JX),JX=1,IFILL-1)
         WRITE (IWR,1001)
      END IF
```

are used to write and echo the last binary output record. The end of view is indicated in this set of 200 array elements by the value 9999 in array I1. The block IF is used to test the echo flag; and, if it is set, to print the shot line data before the end of the view occurs and to print an end of echo message using formatted WRITE statements.

The next statements

```
      WRITE (IWR,1005) MAXT(IFXX)
      NSL = MAXT(IFXM)
      TE = 0.
      JF = 1
      FE = FXCM(1)
      DO 520 I = 1,NTMAX
        TB = TE
        TE = TIMAX(I)
        FB = FE
```

are used to begin assembling the breakdown of critical shot lines by times needed to reach maximum Pk. The first WRITE statement is used to print the number of noncritical shot lines and the heading for shot line breakdown. The next four assignment statements are used to initialize four variables: NSL is used to count the critical shot lines, TE is the end time, JF is the subscript for the flux table, and FE is the end flux level. The DO statement is used to initiate a loop to iterate through each time interval for shot line breakdown, and the next three assignment statements are used to determine begin time, end time, and begin flux level for the Ith time interval.

The statements

```
        JFF = JF
        DO 518 J = JFF,IFM
          JF = J
```

```
              IF (TE .LE. FTIM(J)) GO TO 519
              IF (TE .LE. FTIM(J+1)) THEN
                JF = J + 1
                GO TO 519
              END IF
   518      CONTINUE
            JF = IFMAX
   519      FE = FXCM(JF)
            WRITE (IWR,1006) TB,TE,FB,FE,MAXT(I)
            NSL = NSL + MAXT(I)
   520 CONTINUE
```

are used to find the flux level at the interval end time and to print the shot line breakdown. The DO loop is used to search the FTIM array to obtain the time interval, JF, which includes TE. The assignment statement following Statement 518 is executed only if the end time is greater than all values in the FTIM array. In this case, the last flux level in the distribution is the end flux level. The WRITE statement is used to print the shot line breakdown. The variable NSL is computed in the last assignment statement and is used to count the total number of critical shot lines. Statement 520 is used to end the DO loop which is executed for every shot line time interval.

The statements

```
      TB = TE
      TE = 1.E30
      FB = FE
      FE = FXCM(JFMAX)
      WRITE (IWR,1006) TB,TE,FB,FE,MAXT(IFXM)
      WRITE (IWR,1011) NSL
      NSL = NSL + MAXT(IFXX)
      WRITE (IWR,1012) NSL
```

are used to print a summation of critical shot line breakdowns. The first four assignment statements are used to compute the begin time, end time, begin flux, and end flux for the last interval. The end time, TE, is assigned a value of 10^{30} to imply that this interval applies to all times greater than the last begin time. Two WRITE statements are used next to print the last line of the breakdown table and the total number of shot lines. The last assignment statement is used to add the number of noncritical shot lines, MAXT(IFXX), to the number of critical shot lines, NSL. The last WRITE statement is then used to print the total number of shot lines.

The statements

```
      WRITE (IWR,1009)
      DO 600 J = 1,NOCOMP
        IF (THRAT1(J) .GT. 1.0) THEN
          WRITE (IWR,1010) ICOMP(J),THRAT1(J),THRAT2(J)
        ELSE IF (THRAT2(J) .GT. 1.0) THEN
          WRITE (IWR,1013) ICOMP(J),THRAT2(J)
        END IF
  600 CONTINUE
```

are used to print the last page of QKPK output, which consists of a list of all components having at least one encounter where the LOS thickness is less than that necessary to achieve a non-zero Pk or that necessary to achieve the maximum Pk for a single encounter of the component. In these cases, the component is completely perforated before a non-zero or a maximum Pk value is obtained. The first WRITE statement is used to test the arrays THRAT1 and THRAT2 for each component. The first condition of the block IF statement is used to detect components with a LOS thickness less than that necessary to achieve a non-zero Pk. When such a component is found, the WRITE statement is executed and execution then falls through to the end of the loop. The second condition of the block IF is used to test for a component with a LOS thickness less than that necessary to achieve the maximum Pk for a single encounter of the component. When such a component is found, the last WRITE statement is executed. Execution then continues with Statement 600, the end of the loop. Entries in this table of output indicate possible errors in the selection of DEPTH values to achieve the non-zero Pk and the maximum Pk.

The statements

```
      STOP
  530 CONTINUE
      WRITE (IWR,1008) NENC
      STOP
```

are used to stop the execution of this program. The first STOP statement is used to halt program execution and is the normal end point for this program after completely processing the shot line data for one view. The statements after Statement 530 are executed only if the number of encounters for one shot line exceeds 100. The WRITE statement is used to print an error message. The last STOP statement is used to halt program execution after the fatal error.

The last group of statements

```
 1000 FORMAT ('1 THE FOLLOWING IS AN ECHO OF THE DATA WRITTEN ON THE ',
     $  'QKPK OUTPUT FILE ( LOGICAL UNIT ',I2,' )'/)
 1001 FORMAT ('0END OF ECHO OF DATA WRITTEN ON QKPK OUTPUT FILE')
 1002 FORMAT ('0PKTIME(J,I), PLS(I), I1(I), J=1,10, I=1,200'/
     $  200(11F10.5,I8/))
 1003 FORMAT ('1VALUES USED FOR THIS RUN'//
     $         ' ICOMP     IFG         MAT         ITAB        IANA       TINIT',
     $         5X,'THINFL      RHOF          IY          IU          CARD'/
     $         8X,'DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)',
     $         2X,'PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)'/
     $         8X,'DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)'
     $         2X,'PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)'/)
 1004 FORMAT (I6,I7,3I10,F11.2,2F10.2,I9,I10,I11/2(F15.2,9F10.2/))
 1005 FORMAT ('1 TOTAL NUMBER OF NON-CRITICAL SHOTLINES =',I7//
     $  '0BREAKDOWN OF CRITICAL SHOTLINES BY TIME NEEDED TO REACH ',
     $  'MAXIMUM PK'/'0BEGIN TIME',5X,'END TIME',5X,'BEGIN FLUX',5X,
     $  'END FLUX',5X,'NUMBER OF SHOTLINES')
 1006 FORMAT (1X,F10.2,3X,F10.2,5X,F10.2,3X,F10.2,10X,I10)
 1008 FORMAT ('0NENC =',I10,' WHICH IS GREATER THAN 100'/
     $  '0...PROGRAM HALTING')
 1009 FORMAT ('1 THE FOLLOWING COMPONENTS HAVE AT LEAST ONE SHOTLINE',
     $  ' ENCOUNTER WHERE THE THICKNESS IS LESS THAN PKMN OR PKMX....'/
     $  6X,'IF THE THICKNESS IS LESS THAN PKMN, COMPONENT PK IS ZERO ',
     $  'FOR THAT ENCOUNTER'/6X,'IF THE THICKNESS IS LESS THAN PKMX, '
     $  'COMPONENT PK IS LESS THAN 1.0 FOR THAT ENCOUNTER'/'0COMPONENT',
     $  5X,'MAXIMUM VALUE OF PKMN/THICKNESS',5X,'MAXIMUM VALUE OF ',
     $   'PKMX/THICKNESS')
 1010 FORMAT (1X,I6,20X,F10.3,25X,F10.3)
 1011 FORMAT ('0',25X,'TOTAL NUMBER OF CRITICAL SHOTLINES =',I10)
 1012 FORMAT (/'0TOTAL NUMBER OF SHOTLINES =',I10)
 1013 FORMAT (1X,I6,21X,'LESS THAN 1.0',21X,F10.3)
 1014 FORMAT ('0AZ, EL, GRID, IDVEH, YMAX, YMIN, ZMAX, ZMIN, NOCOMP'/
     $  3F10.5,I8,4F10.5,I8)
 1015 FORMAT ('0ICOMP(I), IFG(I), IY(I), IU(I), NOPNTS(I), (PKVAL(J,I)',
     $  'J=1,NOPNTS(I)),I=1,NOCOMP')
 1016 FORMAT (5I6,10F8.2)
 1017 FORMAT ('0IRVRS, IFMAX, (FTIM(I), FXCM(I), I=1,IFMAX',
     $  2I10/4(2F10.3,5X))
      END
```

is used to define all input and output formats and to end Program QKPK.

SUBROUTINE RAT

This subroutine is used to compute the penetration rate for each encounter along a shot line. Program QKPK calls Subroutine RAT once for every level of flux in the flux distribution. The computed rates are stored in the array RATES, subscripted by encounter number. The penetration rate for a component may be computed by using one of three methods: melt-in-place analysis, melt-removed analysis, or table look-up. The first two methods are used for metallic components and use Subroutine PROPY. The third method uses Subroutine TABLE.

The first set of statements

```
      SUBROUTINE RAT (RATES,IR,MAXENC)
      DIMENSION LOC(100),RATES(MAXENC)
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $            TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $            RHOF(500),INVRS,IU(500),IY(500)
      COMMON    THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON   /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON   /TWO/ XLOS(100),JHT(100),NENC,OHQ(100)
      COMMON   /PROP/ ALP,RHX,CP,TMLT,XLAMBD,RATE,JCOMP,J,DP,XK,TVAP,
     $            CPL,ITER,T
      COMMON   /SIZES/ ITC,IFX,IFXX
```

is used to facilitate transferring data among subroutines and to declare array dimensions. The SUBROUTINE statement is used to accept three arguments: the rates array, the subscript for the flux distribution, and the dimension of the rates array. The DIMENSION statement is used to establish a local integer array LOC and to declare the argument RATES to be a one dimensional array. Note that RATES is a two dimensional array in the calling program, but subroutine RAT can only be used to place values in array positions for a constant flux index IR, the second subscript. Five COMMON statements are used to transfer data and to declare array dimensions.

The statements

```
      IF (IR .EQ. 1) THEN
        IFLAG = 0
        NCRIT = 0
        DO 100 I = 1,NENC
          JCOMP = JHT(I)
          CALL BSRCH(ICOMP,NOCOMP,ITC)
          LOC(I) = J
          IF (IFG(J) .NE. 0) THEN
            NCRIT = NCRIT + 1
            IFLAG = I
          END IF
100     CONTINUE
        NENC = IFLAG
        IF (IFLAG .EQ. 0) RETURN
      END IF
```

are executed once for each shot line. They are used to build the component number link array LOC and to count the number of critical encounters on the shot line. The first block IF statement is used to test the argument IR, which equals 1 the first time Subroutine RAT is called for a shot line. When IR is greater than 1, the block IF is bypassed; otherwise, the DO loop is executed for each encounter on the shot line. The encounter component number is assigned to JCOMP. Subroutine BSRCH is invoked to search the component number array, ICOMP, to find the subscript J for the component number matching JCOMP. The assignment statement following the CALL statement is used to store the subscript J in the LOC array. The next block IF statement determines if the encounter is with a critical component. If not, execution continues at the end of the loop; otherwise, the two assignment statements in the THEN branch are used to count the number of critical components on the shot line, NCRIT, and to save the encounter number of the last critical encounter in IFLAG. After the DO loop is completed, the next assignment statement is used to reassign the number of components, NENC, so that any noncritical encounters at the end of the shot line are excluded. The IF statement is used to return to Program QKPK if there are no critical encounters on the shot line. In this case, the variable IFLAG is also tested in Program QKPK so that this subroutine will not be invoked again for the same shot line.

The statements

```
DO 300 I = 1,NENC
  J = LOC(I)
  DP = XLOS(I)
```

are used to initiate a DO loop which will iterate for every encounter. The first assignment statement is used to get the subscript, J, for the component information arrays from the LOC array. The next assignment statement is used to set DP equal to the component LOS thickness.

The next statements

```
IF (IR .EQ. 1) THEN
  IF (THINFL(J) .GT. 0.) THEN
    DP = THINFL(J) * OBQ(I)
  ELSE
    DP = XLOS(I) * RHOF(J)
  END IF
  XLOS(I) = DP
END IF
WATE = 0.
```

are used to compute the actual component thickness for the encounter. The first block IF statement is used to bypass these steps if the XLOS values have already been computed for this shot line. The next block IF statement is used to determine which method to use in computing the actual thickness. If the influence mode thickness is greater than 0.0, the THEN branch of the block IF will be used to compute the actual thickness as the product of the wall thickness and the secant of the obliquity angle. This value (in meters) is assigned to DP (the variable THINFL was converted to meters in Program QKPK). If the other method is used, the ELSE branch of the block IF will be used to compute the product of the LOS thickness (converted to meters in Program QKPK) and the input density factor of that component. The actual thickness for the Ith encounter is stored in the XLOS array, using the assignment statement following the end of the second block IF. The last assignment is used to initialize the variable to be zero.

The statements

```
IF (DP .GT. 0.) THEN
  IF (ITAB(J) .EQ. 0) THEN
```

are used to determine the appropriate method for computation of the penetration rate. If the component thickness is less than or equal to 0.0, the THEN branch of the first block IF is not executed, thereby bypassing all rate computations and resulting in a rate equal to 0.00. The second block IF statement determines whether a table look-up code has been specified. If not, the THEN branch is executed; otherwise, the ELSE branch is used.

The next statements

```
- ITER = 0
  CALL PROPY
  CPS = CP
  RHO = RHX
```

are used to begin a rate computation, using one of the analysis types. The first two statements are used to call Subroutine PROPY with the iteration flag equal to 0. Subroutine PROPY is used to compute the material dependent properties--melting temperature, vapor temperature, heat of fusion, coupling coefficient, and density--and the temperature dependent properties--thermal conductivity and specific heat. The last two assignment statements are used to transfer the specific heat value to variable CPS and the density value to variable RHO.

The statements

```
          IF (IANA(J) .EQ. 1) THEN
            ITER = 1
            T = TMLT
335         CONTINUE
              CALL PROPY
              TSTAR = TMLT + FLX * ALP * DP / XK
            IF (ABS(T-TSTAR) .GE. 100.) THEN
              T = TSTAR
              GO TO 335
            END IF
```

are used to compute the component's temperature rise for the melt in place analysis mode. The first block IF statement determines whether the melt in place analysis type is specified. If so, the statements in the THEN branch are executed; otherwise, the ELSE branch is executed. The next two assignment statements are used to set the iteration flag, ITER, equal to 1 and the temperature, T, equal to the melting temperature. The rest of the statements form a loop which is used iteratively to compute the limit of the temperature rise. Since the iteration flag ITER equals 1, Subroutine PROPY recomputes the temperature dependent properties--thermal conductivity, XK, and specific heat, CP. The assignment statement following the CALL statement is used to compute the new temperature TSTAR. The mathematical expression for computing TSTAR is:

$$\boxed{T^* = T_m + \frac{\alpha F Z}{k}} \qquad (3\text{-}12)$$

The second block IF is used to branch out of the loop when the temperature rise, i.e. the change in temperature, is less than 100 from the previous loop.

The next statements

```
      IF ((TSTAR .GT. TVAP) .AND. (TVAP .GT. 0.)) THEN
        STAR = XK * (TVAP - TMLT) / UP
        RATE = STAR / (RHO * (CPS * (TMLT - TINIT(J)) +
$         XLAMBD + CP * .5 * (TVAP - TMLT)))
      ELSE
        RATE = FLX * ALP / (RHO * (CPS * (TMLT - TINIT(J)) +
$         XLAMBD + CP * (.5 * (TSTAR + TMLT) - TMLT)))
      END IF
```

are used to compute the penetration rate for the melt in place analysis type. The block IF statement is used to determine which of two melt in place rate equations should be used. If the peak temperature is greater than the vapor temperature and the vapor temperature is positive, the rate equation in the THEN branch will be used. This equation in mathematical form is:

$$\boxed{R = \frac{k\,\dfrac{(T_v - T_m)}{Z}}{\rho\left[\text{Cps }(T_m - T_1) + \lambda + \text{Cp}\ell\left(\dfrac{T_v - T_m}{2}\right)\right]}} \qquad (3\text{-}16)$$

If the peak temperature does not exceed the vapor temperature or the material type has a vapor temperature of 0, the second melt in place rate equation in the ELSE branch will be used. The mathematical form of the second rate equation is:

$$\boxed{R = \frac{\alpha F}{\rho\left[\text{Cps }(T_m - T_1) + \lambda + \text{Cp}\ell\left(\dfrac{T^* + T_m}{2} - T_m\right)\right]}} \qquad (3\text{-}10)$$

The statements

```
      ELSE
        RATE = FLX * ALP / (RHO * (CP * (TMLT - TINIT(J)) +
$         XLAMBD))
      END IF
```

are used to compute the penetration rate for the melt removed analysis type. This rate equation is written mathematically as:

$$\boxed{R = \frac{\alpha F}{\rho\left[\text{Cps }(T_m - T_1) + \lambda\right]}} \qquad (3\text{-}7)$$

The statements

```
        ELSE
          CALL TABLE
        END IF
      END IF
```

invoke Subroutine TABLE when a look-up code has been specified to compute a penetration rate for nonmetallic components.

The statements

```
RATES(I) = RATE
JHT(I) = J
```

are used to store the penetration rate and to reassign the component link. The first assignment statement is used to store the computed penetration rate in the array, RATES, indexed by the encounter number. The second assignment statement is used to store the component information array's subscript J in the JHT array. This array previously contained the component identification numbers.

The final group of statements

```
300 CONTINUE
    RETURN
    END
```

contains Statement 300, the last statement in the encounter loop. After a penetration rate has been computed and stored for every encounter on the shot line, the RETURN statement will be used to relinquish control to Program QKPK. The END statement is used to conclude this program unit.

SUBROUTINE RDATA

This subroutine is used by Program QKPK to read the formatted input deck. The data in this deck include: number of components, shot line reverse flag, units conversion factor for the input, flux distribution table, and, optionally, time intervals for shot line breakdown. Subroutine RDATA is also used to print the flux distribution table, the first page of output from Program QKPK, and to test the flux values and the number of components. An error detected by these tests will halt program execution. The same tests are made by Program PRERD and any error should be corrected before proceeding to Program QKPK.

The statements

```
      SUBROUTINE RDATA
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IAMA(500),
     $         TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $        RHOF(500),IRVRS,IU(500),IY(500)
      COMMON  THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON /TWO/ XLOS(100),JHT(100),NENC,OHQ(100)
      COMMON /FOUR/ RATES(100,25),FTIM(25),FXCM(25),FXM(25),IFMAX,
     $        TIMAX(25),MAXT(27),NTMAX
      COMMON /LUNITS/ IRD,IWR,IIN,IOUT
      COMMON /SIZES/ ITC,IFX,IFXX
```

are used to establish array dimensions and common storage allocations. Data for variables are transferred to and from other program units by using the common storage areas.

The next statements

```
      READ (IRD,1000) NOCOMP
      IF (ITC .GE. NOCOMP+2) THEN
        READ (IRD,1000) IRVRS,CINCH
        READ (IRD,1000) IFMAX
        READ (IRD,1002) (FXCM(I),I=1,IFMAX)
        READ (IRD,1002) (FTIM(I),I=1,IFMAX)
```

are used to read the number of components, NOCOMP, and, if the value read is not too large, to read the shot line reverse flag, the input unit conversion factor, and the flux table. If NOCOMP is too large, the ELSE branch of the block IF is executed.

The statements

```
        WRITE (IWR,2000) NOCOMP,IRVRS,CINCH
        WRITE (IWR,2001) IFMAX
        TE = 0.
        DO 110 I = 1,IFMAX
          TA = TE
          TE = FTIM(I)
          WRITE (IWR,2002) TA,TE,FXCM(I)
110     CONTINUE
        TA = TE
        TE = 1.E30
        WRITE (IWR,2002) TA,TE,FXCM(IFMAX)
```

are used to print the first page of QKPK output. The first two WRITE statements are used to print the number of components, the shot line reverse flag, the input unit conversion factor, and the number of points in the flux distribution. The DO loop is used to print the flux distribution table in a format that shows a flux level, FXCM(I), lasting for a time interval, TB to

TE. Since the last flux level applies to all times greater than the last time in array FTIM, an extra line is printed in the flux table with an end time value of 10^{30}. This value requires more digits than the format for the last WRITE statement will allow, so the end time for this flux level is printed as a row of asterisks, symbolizing an infinite end time.

The statements

```
        DO 160 I = 1,IFMAX
          IF ((FXCM(I) .GT. 0.) .AND. (FXCM(I) .LE. 6.0E+04)) THEN
            FXM(I) = FXCM(I) * 10000.
          ELSE
            WRITE (IWR,100) FXCM(I)
            STOP
          END IF
 160    CONTINUE
```

are used to check if each flux level is within the acceptable range (0.0 to 60000.0 watts/cm^2) and to convert the flux values to another unit, watts/m^2. The DO loop is used to cycle through each of the IFMAX flux levels. The block IF statement is used to test the flux level. If the flux level passes the test, the flux value is converted to watts/m^2 and is stored in array FXM. If the flux level is too low or too high, the WRITE statement is used to print a warning message and the program stops.

The statements

```
        DO 40 I = 1,NOCOMP
          READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $      IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
          IF (IOS .EQ. 0) THEN
            IF ((NOPNTS(I) .LT. 2) .OR. (NOPNTS(I) .GT. 10)) THEN
              WRITE (IWR,1010) I,NOPNTS(I)
              STOP
            ELSE IF (NOPNTS(I) .GT. 4) THEN
              READ (IRD,1001,IOSTAT=IOS)
            END IF
          END IF
          IF (IOS .GT. 0) THEN
            WRITE (IWR,1015) I
            WRITE (IWR,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $        IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
            STOP
          ELSE IF (IOS .LT. 0) THEN
            WRITE (IWR,1020,IOSTAT=IOS) I-1,NOCOMP
            STOP
          END IF
 40     CONTINUE
```

are used to read the component information cards by executing a DO loop for every component. Each component requires two to three cards depending on the number of points at which probabilities of kill are to be established. The READ statement sets a status indicator IOS to report the outcome of the READ operation. If IOS equals 0, the READ executed normally and the value of NOPNTS(I) is checked to insure that it is within the subscript range of the DEPTH and PKVAL arrays and that there are at least two elements. If the subscript number fails these criteria, an error message is printed and the program terminates. If NOPNTS(I) meets these conditions, an additional read is performed if more than four P(K/H) points are being entered so as to bypass this third component card. If IOS is greater than 0, an error occurred during one of the two possible reads. An error message is printed along with the values that were successfully read. The program then terminates. If IOS is less than 0, an unexpected end of file occurred. The program assumes that too large a value of NOCOMP was entered, although a missing component card could cause the same effect. A messag is printed and the program terminates.

The statements

```
      REWIND (IRD)
      DO 45 I = 1,5
        READ(IRD,1001)
   45 CONTINUE
```

reset the formatted data file for reading. After setting the file pointer to the first record, the first five records are bypassed using a DO loop since they have been previously checked for content errors.

The next statements

```
      DO 50 I = 1,NOCOMP
      READ (IRD,1001) ICOMP(I),MAT(I),IFG(I),ITAH(I),IANA(I),
     $   TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I),(DEPTH(J,I),
     $   PKVAL(J,I), J = 1,NOPNTS(I))
   50 CONTINUE
```

are used to read the component information data by executing the DO loop once for every component. Each iteration of the loop causes the READ statement to read information on one component from two or three data cards, depending on the number of points for which probabilities are being entered. These data are stored in the component information arrays subscripted by I, the DO loop index.

The statements

```
      READ (IRD,1000,END=60) NTMAX
      READ (IRD,1002) (TIMAX(I),I=1,NTMAX)
   60 CONTINUE
```

are used to read the optional shot line breakdown time intervals. If these data are not included, an end of data will be detected, which will cause a branch to Statement 60. Otherwise, the first READ statement will be used to accept the number of time bins and the second READ statement will be used to read each of the values for array TIMAX by executing an implied DO loop.

The statements

```
      ELSE
        WRITE (IWR,2004) NOCOMP
        STOP
```

form the ELSE branch of the block IF when the number of components, NOCOMP, is too large. The WRITE statement prints an error message and then the program stops.

The statements

```
      END IF
      RETURN
```

close the block IF and return control to the calling program.

The last group of statements

```
  100 FORMAT(/5X,'***** INPUT ERROR ***** FLUX =',E10.2/
     $        'PROGRAM HALTING')
 1000 FORMAT (I5,2E10.2)
 1001 FORMAT (I5,I3,1X,I1,I2,13X,I1,F5.0,32X,F3.2,7X,F3.2,2I2/
     $        I5,9F7.4/11F7.4)
 1002 FORMAT (10F7.0)
 1010 FORMAT ('1',5X,'***** INPUT ERROR *****     AN INVALID NUMBER ',
     $         'FOR THE VARIABLE NOPNTS(I) WAS READ:'
     $         /28X,'I = ',I4,' NOPNTS(I) = ',I4)
 1015 FORMAT (/5x,'***** INPUT ERROR *****     ERROR OCCURRED DURING ',
     $         'READ OF COMPONENT CARD #',I4,' IN THE SEQUENCE'/
     $          28X,'THE FOLLOWING VALUES WERE READ:')
 1020 FORMAT (/5x,'***** INPUT ERROR *****     END OF FILE OCCURRED '
     $         'AFTER ',I4,' CARDS WERE READ,'
     $         /28X,'ALTHOUGH NOCOMP = ',I4)
 2000 FORMAT (' NUMBER OF COMPONENTS =',I5,5X,
     $        'IRVRS=',I5,5X,'FACTOR FOR CONVERSION TO INCHES=',E10.4)
 2001 FORMAT (' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5/
     $         '     BEGIN TIME',6X,'END TIME',4X,'FLUX')
 2002 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
 2004 FORMAT ('NOCOMP =',I10,' IS TOO LARGE FOR ARRAY DIMENSIONS'/
     $  '...PROGRAM HALTING')
      END
```

is used to define all input and output formats required by this subroutine.

Intentionally Left Blank

SUBROUTINE READIN

This subroutine is used to read all input required for Program PRERD. This includes the flux distribution table, the component information cards, and, optionally, the time intervals for shot line breakdown. Subroutine READIN is also used to print the first page of PRERD output and to insure that certain input values are within valid ranges.

The statements

```
      SUBROUTINE READIN
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $          TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $          RHOF(500),IU(500),IY(500)
      COMMON    THINFL(500),NOCOMP
      COMMON   /LUNITS/ IRD,IWR
      COMMON   /SIZES/ ITC,IFX
      DIMENSION FXCM(25),FTIM(25),TIMAX(25)
```

are used to declare array dimensions and enable transfer of information to and from this subroutine, using two labeled commons and one unlabeled common.

The statements

```
      READ (IRD,1000) NOCOMP
      IF (ITC .GE. NOCOMP+2) THEN
        READ (IRD,1000) IRVRS,CINCH
        READ (IRD,1001) IFMAX
        WRITE (IWR,2000) NOCOMP,IRVRS,CINCH
        IF ((IFMAX .LT. 1) .OR. (IFMAX .GT. IFX))
     $    WRITE (IWR,101) IFMAX
        WRITE (IWR,2001) IFMAX
        IF (IFMAX .GT. IFX) IFMAX = IFX
```

are used to read the number of target components and to insure that the number of components (plus two for default components) does not exceed the component array's dimensions, ITC. If this input error occurs, the ELSE branch of the block IF statement is executed where a warning message is printed and the program halts. This is the first of four fatal input errors that can be detected by Subroutine READIN. If the error does not occur, the subroutine continues by reading the shot line reverse flag, the input units conversion factor, and the number of points in the flux distribution table. This information is then echoed to device, IWR, in a formatted WRITE statement. If the number of points in the flux distribution, IFMAX, is negative or exceeds the array dimensions, an error message is printed but the subroutine continues by writing the heading for the flux table. The last IF statement insures that IFMAX is less than or equal to the flux array dimensions.

The next set of statements

```
        READ (IRD,1002) (FXCM(I),I=1,IFMAX)
        READ (IRD,1002) (FTIM(I),I=1,IFMAX)
        TE = 0.
        DO 110 I = 1,IFMAX
          TB = TE
          TE = FTIM(I)
          WRITE (IWR,2002) TB,TE,FXCM(I)
  110   CONTINUE
        TB = TE
        TE = 1.E30
        WRITE (IWR,2002) TB,TE,FXCM(IFMAX)
```

is used to read and write the flux distribution table. The flux levels apply to time intervals, TB to TE. Note that the subroutine prints one more point than it reads in the flux distribution. The last interval printed is an extra point and is used to indicate that the last flux level applies to all times greater than the last FTIM value.

The statements

```
      DO 160 I = 1,IFMAX
        IF ((FXCM(I) .LE. 0.) .OR. (FXCM(I) .GT. 6.0E+04))
     $     WRITE (IWR,100) FXCM(I)
160   CONTINUE
```

are used to test each element of array FXCM to insure that the values are within the limits, 0.0 through 60000.0. If a flux level is outside of these bounds, an error message is printed but the subroutine continues.

The statements

```
      DO 40 I = 1,NOCOMP
        READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $     IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
        IF (IOS .EQ. 0) THEN
          IF ((NOPNTS(I) .LE. 0) .OR. (NOPNTS(I) .GT. 10)) THEN
            WRITE (IWR,1010) I,NOPNTS(I)
            STOP
          ELSE IF (NOPNTS(I) .GT. 4) THEN
            READ (IRD,1001,IOSTAT=IOS)
          END IF
        END IF
        IF (IOS .GT. 0) THEN
          WRITE (IWR,1015) I
          WRITE (IWR,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $       IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
          STOP
        ELSE IF (IOS .LT. 0) THEN
          WRITE (IWR,1020,IOSTAT=IOS) I-1,NOCOMP
          NOCOMP = I - 1
        END IF
40    CONTINUE
```

are used to read the component information cards by executing a DO loop for every component. Each component requires two to three cards depending on the number of points at which probabilities of kill are to be established. The READ statement sets a status indicator IOS to report the outcome of the READ operation. If IOS equals 0, the READ executed normally and the value of NOPNTS(I) is checked to insure that it is within the subscript range of the DEPTH and PKVAL arrays. If the subscript is out of range, an error message is printed and the program terminates. This is the second fatal error detected by this subroutine. If NOPNTS(I) is within the array bounds 1 to 10, an additional read is performed if more than four P(K/H) points are being entered so as to bypass this third component card. If IOS is greater than 0, an error occurred during one of the two possible reads. An error message is printed along with the values that were successfully read. The program then terminates. This is the third fatal error detected by this subroutine. If IOS is less than 0, an unexpected end of file occurred. The program assumes that too large a value of NOCOMP was entered, although a missing component card could cause the same effect. A message is printed and execution continues after reducing the value of NOCOMP.

The statements

```
        REWIND (IRD)
        DO 45 I = 1,5
          READ(IRD,1001)
   45   CONTINUE
```

reset the formatted data file for reading. After setting the file pointer to the first record, the first five records are bypassed using a DO loop since they have been previously checked for content errors.

The statements

```
        DO 50 I = 1,NOCOMP
        READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $    IANA(I),IINTI(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I),
     $    (DEPTH(J,I),PKVAL(J,I),J = 1,NOPNTS(I))
   50   CONTINUE
```

are used to read the component information cards by executing a DO loop for every component. These data are stored in several component information arrays which are defined at the end of this section in the List of Abbreviations and Symbols. The user may place the component information cards in any order so long as each component's continuation card(s) follow the initial card in the proper sequence.

The following statements

```
        READ (IRD,1000,END=60) NTMAX
        IF ((NTMAX .GE. 1) .AND. (NTMAX .LE. IFX)) THEN
          READ (IRD,1002) (TIMAX(I),I=1,NTMAX)
          WRITE (IWR,1005) (TIMAX(I),I=1,NTMAX)
          RETURN
        ELSE
          WRITE (IWR,102) NTMAX
          WRITE (IWR,1007)
          STOP
        END IF
```

are used to read the time intervals for shot line breakdown if the user includes this optional input. This is done by first reading NTMAX, the number of points in the time interval array. The value of NTMAX is then tested and if it is within the array limits, the subroutine reads and echoes to device IWR the upper boundary times for which Pk's and vulnerable areas are to be computed. Control then returns to the calling program. If NTMAX is outside the array bounds, an error message is printed and execution stops. This is the fourth fatal error detected by this subroutine. If the user omits the breakdown time intervals, the first READ statement detects an end of file condition on device IRD and branches to Statement 60.

The statements

```
        ELSE
          WRITE (IWR,1007)
          STOP
        END IF
```

are executed only if NOCOMP, the number of components, exceeds the array dimensions. These statements are used to print a warning message and stop program execution.

The statements

```
   60 CONTINUE
      WRITE (IWR,1006)
      RETURN
```

are executed when the read statement following Statement 50 detects an end of file condition. In this event, the time intervals for the flux distribution will be used as the breakdown time intervals. The statements listed after Statement 60 are used to print this information in a warning message and then return control to the calling program.

The statements

```
  100 FORMAT(/5X,"***** INPUT ERROR ***** FLUX =",E10.2)
  101 FORMAT(/5X,"***** INPUT ERROR ***** IFMAX =",I10)
  102 FORMAT(/5X,"***** INPUT ERROR ***** NTMAX =",I10)
 1000 FORMAT(I5,2F10.2)
 1001 FORMAT(I5,I3,1X,I1,I2,13X,I1,F5.0,32X,F3.2,7X,F3.2,2I2/
     $   I5,9F7.4/11F7.4)
 1002 FORMAT (10F7.0)
 1005 FORMAT (' TIMAX VALUES ='/(10F10.2))
 1006 FORMAT (' NO TIMAX VALUES WERE READ IN ... FLUX TIME POINTS ',
     $  'WILL BE USED')
 1007 FORMAT (' ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING')
 1010 FORMAT ('1',5X,'***** INPUT ERROR *****     AN INVALID NUMBER ',
     $         'FOR THE VARIABLE NOPNTS(I) WAS READ:'
     $         /28X,'I = ',I4,' NOPNTS(I) = ',I4)
 1015 FORMAT (/5X,'***** INPUT ERROR *****     ERROR OCCURRED DURING ',
     $         'READ OF COMPONENT CARD #',I4,' IN THE SEQUENCE'/
     $         28X,'THE FOLLOWING VALUES WERE READ:')
 1020 FORMAT (/5X,'***** INPUT ERROR *****     END OF FILE OCCURRED '
     $         'AFTER ',I4,' CARDS WERE READ,'
     $         /28X,'ALTHOUGH NOCOMP = ',I4)
 2000 FORMAT(' NUMBER OF COMPONENTS =',I5,5X,
     $    ' IRVRS =',I5,5X,'FACTOR FOR CONVERSION TO INCHES =',F10.4)
 2001 FORMAT (/' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5//
     $  ' BEGIN TIME',5X,'END TIME',7X,'FLUX')
 2002 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
      END
```

are used to specify all formats for the READ and WRITE statements in this subroutine and to end this program unit.

SUBROUTINE REVRSE

When using the QKLOOK programs, a user may obtain data for a second view without generating a new binary LOS file. This is done by setting the shot line reverse flag, IRVRS, on the second QKPK input data card equal to 1. This will cause Program QKPK to call Subroutine REVRSE for every shot line in the viewing plane. This subroutine is used to reverse the order of the component encounters along the shot line and to transform the shot line coordinates to the new system. The resulting shot line is the same as the orginal one but in the opposite direction. Note that LOS data from Program MAGIC does not include exit obliquity angles. If MAGIC data is used, the obliquity angles on the reversed shot line are set equal to the original obliquity angles.

The statements

```
      SUBROUTINE REVRSE(J)
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $          TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $          RHOF(500),IRVRS,ID(500),IY(500)
      COMMON  THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /TWO/ XLOS(100),JHT(100),NENC,OBQ(100)
```

are used to specify COMMON blocks and one argument, which are required for transfer of information to and from this subroutine. The SH array is the only one required from the unlabeled COMMON block. COMMON /TWO/ contains the number of encounters NENC, and the encounter arrays which will be reversed by this subroutine. The argument J is the shot line index needed as a subscript for array SH.

The statement

```
      SH(1,J) = -SH(1,J)
```

is used to reflect the y-coordinate of the shot line about the Z-axis while the z-coordinate of the shot line remains the same.

The next statements

```
      DO 30 N = 1,NENC/2
        JJ = NENC - N + 1
        J1 = JHT(N)
        S1 = OBQ(N)
        S2 = XLOS(N)
        JHT(N) = JHT(JJ)
        OBQ(N) = OBQ(JJ)
        XLOS(N) = XLOS(JJ)
        JHT(JJ) = J1
        OBQ(JJ) = S1
        XLOS(JJ) = S2
   30 CONTINUE
      RETURN
      END
```

are used to reverse the order of the components along the shot line. There are three encounter information arrays which must be interchanged. The DO loop is used to execute the interchange NENC/2 times. The variable JJ is computed in the first assignment statement and is the subscript of the array element to be interchanged with the Nth array element. The last nine assignment statements in the DO loop are used to interchange the Nth and JJth elements in the three encounter information arrays.

SUBROUTINE TABLE

This subroutine is called by Subroutine RAT and is used to compute a penetration rate for an encounter with a nonmetallic component using a table look-up procedure. The table look-up code is defined by the user in the formatted component information data.

The first set of statements

```
      SUBROUTINE TABLE
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $         TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $         RHOF(500),IRVRS,IU(500),IY(500)
      COMMON  THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON /PROP/ ALP,RHO,CP,TMLT,XLAMBD,RATE,JCOMP,J,DP,XK,TVAP,
     $         CPL,ITER,T
      COMMON /LUNITS/ IRD,IWR,IIN,IOUT
```

is used to facilitate transfer of data to and from this subroutine and to declare array dimensions. The ITAB array in the unlabeled COMMON contains the table look-up codes. The variable FLX in COMMON /ONE/ is the flux level for this rate computation. COMMON /PROP/ contains the variable, J, which is the component index used for the table look-up code, and the variable, RATE, which is the penetration rate computed in Subroutine TABLE.

The statements

```
      IF (ITAB(J) .EQ. 1) THEN
C
C         PLEXIGLASS
C
        RATE = 4.35E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 2) THEN
C
C         TRIPLEX DIELECTRIC
C
        RATE = 2.6822E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 3) THEN
C
C         FIBRITE
C
        RATE = 2.633E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 4) THEN
C
C         RUBBER
C
        RATE = 2.6909E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 5) THEN
C
C         GLASS
C
        RATE = .35E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 6) THEN
C
C         FIBER GLASS
C
        RATE = .44E-10 * FLX * 0.9
```

are part of an IF block which takes the appropriate action based on the value of the table look-up code, where ITAB(J) is the table look-up code for the component being encountered. For acceptable values, the penetration rate for the material specified, is computed as a product of the material coefficient, the flux level FLX, and an absorption factor of 0.9. This is written mathematically as:

$$\boxed{R = 0.9CF} \tag{3-1}$$

The next statements

```
      ELSE IF (ITAB(J) .EQ. 7) THEN
C
C             PYRUCERAM
C
        IF (FLX .GT. 3.74E+6) THEN
          RATE = -1.4E-4 + 3.74E-11 * FLX * 0.9
        ELSE
          WRITE (IWR,2000)
          STOP
        END IF
```

are used to compute the penetration rate when the table look-up code equals 7. For this table look-up code, the flux level must be greater than 374 watts/cm^2. When the flux level is high enough, the assignment statement in the THEN branch of the IF block is used to compute the penetration rate. Otherwise, the ELSE branch prints a fatal error message and the program terminates.

The statements

```
      ELSE IF (ITAB(J) .EQ. 8) THEN
C
C             GLASS FIBER EPOXY
C
        RATE = 0.556E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 9) THEN
C
C             GRAPHITE EPOXY
C
        RATE = 0.119E-10 * FLX * 0.9
```

are used to compute the penetration rate when the table look-up code equals 8 or 9.

The statements

```
      ELSE
        RATE = 0.0
        WRITE(IWR,1000) JCOMP,ITAB(J)
      END IF
      RETURN
```

are used to assign a penetration rate equal to 0.0 and to write a warning message when the table look-up code does not have an acceptable value. The END IF statement closes the block IF and the RETURN statement returns control to Subroutine RAT.

The final group of statements

```
 1000 FORMAT(' THIS COMPONENT DOES NOT FIT THE BOUNDS FOR TABLES',2I6)
 2000 FORMAT(' THE FLUX IS INDICATED TO BE LESS THAN 374 W/CM**2'/
     $        '0CHECK SUBROUTINE TABLE...PROGRAM HALTING')
      END
```

is used to define the formats for the error message output and to end Subroutine TABLE.

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PEAKAY (including Subroutine PENT)

Abbreviation or symbol	Equivalent in Mathematical Model	Definition	Units
ASQF	---	Component presented area	ft^2
AVFLAG	---	Component vulnerable area flag; = 0: incremental vulnerable areas = 1: true vulnerable areas	---
AZ	---	Attack azimuth angle	degrees
COMPAV(I,J) $I \leq 500$, $J \leq 10$	A_V	Vulnerable area for the Ith component during the Jth time interval	ft^2 or m^2
EL	---	Attack elevation angle	degrees
FRV	---	Decimal form of the percent of vulnerable area	---
FTIM(I) $I \leq 25$	---	Time arguments for the flux distribution	seconds
FXCM(I) $I \leq 25$	---	Weapon intensity, flux at time FTIM(I)	watts/cm^2
GRID	A_G	Grid cell size--read as a length, converted to ft^2	varies
I	---	In Subroutine PENT, the subscript for the component information arrays corresponding to the encountered component	---
I1(I) $I \leq 200$	---	Number of encounters in each new shot line when the Ith set of arrays are defining a new shot line; otherwise the component numbe⁻ in the encounter with penetration times in R1(J,I) and R3(I) (equals 9999 to indicate end of view)	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
ICOMP(I) $I \leq 500$	---	Component identification number for the Ith component	---
IDVEH	---	Not used, variable on first record of LOS file	---
IFG(I) $I \leq 500$	---	Criticality flag of the Ith component; = 0: noncritical component; ≠ 0: critical component	---
IFMAX	---	Number of points in the flux distribution table	---
II	---	Shot line encounter number in Subroutine PENT	---
IIN	---	Device number for the binary LOS input file (= 2)	---
IOUT	---	Device number for the four binary output files (= 3)	---
IRD	---	Device number for the formatted input file (= 5)	---
IRVRS	---	Shot line reverse flag; = 0: penetration times were computed with encounters in normal order; = 1: penetration times were computed with encounters in reversed order	
ISET(I) $I \leq 10$	---	= 0: the Ith system has no components on the shot line; = 1: the Ith system has at least one component on the shot line	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
IST	---	Equals 0 when the LOS data processing loop is processing data for a new shot line; otherwise, equal to number of the shot line	---
ITC	---	Dimension of the component information arrays (= 500)	---
ITOTL	---	Number of shot lines for the view	---
IU(I) $I \leq 500$	---	Vulnerability flag for Ith component; = 0: singly vulnerable component = 1; multiply vulnerable component	---
IWR	---	Device number for the formatted output file (= 6)	---
IX	---	Index to determine Pk value range in Subroutine PENT	---
IXSET	---	Logical variable assigned a value of .TRUE. when Pk value range found in Subroutine PENT	---
IY(I) $I \leq 500$	---	System number for the Ith component	---
JQQ	---	Number of components with zero kill probability	---
K	---	Index to find flux begin time	---
L	---	System number for a component in Subroutine PENT	---
LL	---	Shot line encounter number in Subroutine PENT	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
LM	---	A previous encounter number on a shot line in Subroutine PENT	---
LOC(I) $I \leq 100$	---	Component number for the Ith shot line encounter in Subroutine PENT (equivalent to NAME(J) in Program PEAKAY)	---
MAXTIM	---	Two plus the number of time intervals NTIME	---
MULSYS	---	Logical variable assigned a value of .TRUE. only if the component is in a system and it is a multiply vulnerable component	---
NAME(I) $I \leq 100$	---	Component number of the Ith shot line encounter	---
NBEG	---	Next shot line encounter number beginning a new encounter loop in Subroutine PENT	---
NENC	---	Total number of encounters on a shot line	---
NEND	---	Number of the last encounter for a time interval	---
NOCOMP	---	Number of components in the target model	---
NOPNTS(I) $I \leq 500$	---	Number of Pk values for Ith component	---
NSHT	---	The viewing plane shot line number	---
NTIM	---	Number of the time intervals in Subroutine PENT	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program PEAKAY (Cont'd.)

Abbreviation or symbol	Equivalent in Mathematical Model	Definition	Units
NTIME	---	Number of time intervals at which presented and vulnerable areas are to be computed	---
NUM	---	Number of encounters with penetration times stored in time arrays for one shot line	---
PAREA(I) $I \leq 500$	A_p	Presented area for the Ith component	varies
PCV	---	Percent of vulnerable area used for one of the component vulnerable area summaries	---
PK	P_i, Pe_i	Component probability of kill from one encounter, then converted to component probability of kill on one encounter and survival on all previous encounters	---
PK1	---	Time at which the maximum possible Pk for a component will be achieved	seconds
PK2	---	Minimum time needed to achieve a nonzero Pk for a component in a time interval	seconds
PK3	t_M	Time needed to achieve the maximum Pk for a component in a time interval.	seconds
PKC(I) $I \leq 100$	---	Component probability of kill for the first through Ith encounters on a shot line	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
PKNK(I) $I \leq 500$	---	Component number for the Ith component with Pk = 0.0 for all shot lines	---
PKS(I) $I \leq 10$	---	Probability of kill for the Ith system of singly vulnerable components	---
PKTIME(J,I) $J \leq 10$, $I \leq 100$	t_N	Time required to penetrate from the beginning of a shot line to the depth for a Pk = PKVAL(J,I) for the Ith shot line encounter	seconds
PKVAL(J,I) $J \leq 10$, $I \leq 100$	Pk_N	Pk associated with penetration to a specified depth for the Ith component	---
PLS(I) $I \leq 100$	t_L	Time required to penetrate from the beginning of a shot line to the depth necessary to perforate the Ith shot line encounter	seconds
PMULT(I,J) $I \leq 10$, $J \leq 10$	---	System probability of kill for the singly vulnerable components in system I at the end of the Jth time interval	---
PMX	---	Shot line probability of kill excluding multiply vulnerable components	---
PRNT(1,J) $J \leq 500$	---	Shot line y-coordinate on the viewing plane for an encounter with the Jth component	inches
PRNT(2,J) $J \leq 500$	---	Shot line z-coordinate on the viewing plane for an encounter with the Jth component	inches

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
PRNT(3,J) $J \leq 500$	---	Maximum probability of kill of all shot line encounters with the Jth component, maximum of PKC(II)	---
PRNT(4,J) $J \leq 500$	---	Minimum time to achieve PRNT(3,J) for the Jth component	seconds
R1(J,I) $J \leq 10$, $I \leq 100$	---	Shot line y-coordinate on the viewing plane when the Ith set of array elements are defining a new shot line--R1(1,J); otherwise, the time needed to penetrate to the depth for a Pk = PKVAL(J,I) for the encounter	---
R3(I) $I \leq 200$	---	When the Ith set of array elements are defining a new shot line, R3(I) equals 0.0; otherwise, the minimum time needed to perforate the encountered component	--- or seconds
RVRS	---	Shot line reverse flag, real form of IRVRS: = 0.0; normal shot line = 1.0; reversed shot line	---
SAREA(I) $I \leq 10$	---	Presented area for the Ith system of components	varies
SHW(1,J) $J \leq 200$	---	y-coordinate on the viewing plane of the Jth shot line	inches
SHW(2,J) $J \leq 200$	---	z-coordinate on the viewing plane of the Jth shot line	inches

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program PEAKAY (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
SHW(I,J) $2<I\leq 12$, $J\leq 200$	---	For the (I-2) time interval and the Jth shot line: shot line Pk, excluding multiply vulnerable components; = -9.0: no critical encounters on Jth shot line = -9999.0: end of view indicator	---
SNGSYS	---	Logical variable assigned a value of .TRUE. only if the component is in a system and is singly vulnerable	---
SPK	---	System probability of kill for one system including only singly vulnerable components in the system	---
SQFSQM	---	Conversion factor to convert from square feet to square meters	m^2/ft^2
TB	---	Begin time for a flux distribution interval	seconds
TBEG	---	Begin time of a time interval, time already used in penetrating previous encounters	seconds
TE	---	End time for a flux distribution interval	seconds
TEND	τ	Time since the beginning of weapon flux emission	seconds
TIMES(I) $I\leq 10$	---	Upper time of the Ith time interval and lower time of the (I + 1)st time interval	seconds
TOTL	---	Total target presented area	varies

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PEAKAY (Concluded)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
TVA(I) $I \leq 10$	---	Total target vulnerable area at the Ith time interval	varies
TVAM(I) $I \leq 10$	---	Total target vulnerable area at the Ith time interval	m^2
YMAX	---	Maximum y-coordinate for all shot lines from the viewing plane	inches
YMIN	---	Minimum y-coordinate for all shot lines from the viewing plane	inches
ZMAX	---	Maximum z-coordinate for all shot lines from the viewing plane	inches
ZMIN	---	Minimum z-coordinate for all shot lines from the viewing plane	inches

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PRERD (including subroutines FSORT and READIN)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
CINCH	---	Conversion factor to convert input length units to inches; ≤ 0.0: set conversion factor to 1.0 (units are in inches) ≥ 0.0: use conversion factor (units are not in inches)	inches
DEPTH(J,I) $J \leq 10$, $I \leq 500$	---	Penetration depth of the Ith component to achieve a Pk = PKVAL(J,I)	inches
FTIM(I) $I \leq 25$	---	Time values for the flux distribution table	seconds
FXCM(I) $I \leq 25$	---	Flux, weapon intensity at time FTIM(I)	watts/cm^2
I	---	In Subroutine FSORT, the lower subscript of two compared values	---
IANA(I) $I \leq 500$	---	Analysis type for the Ith component; = 1: melt in place = 2: melt removed	---
IASTER	---	Character variable for an asterisk	---
IBLNK	---	Character variable for a blank	---
ICHK	---	Character variable for a check mark	---
ICOMP(I) $I \leq 500$	---	Component identification number for the Ith component	---
IE(I) $I \leq 34$	---	Array of characters with a blank, asterisk, or check mark for each element	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PRERD (Cont'd.)

Abbreviation or symbol	Equivalent in Mathematical Model	Definition	Units
IFG(I) $I \leq 500$	---	Criticality flag for the Ith component; $= 0$: noncritical $\neq 0$: critical	---
IFMAX	---	Number of points in the flux distribution table	---
IFX	---	Dimension of the flux distribution arrays	---
IM	---	In Subroutine FSORT, the upper subscript of two compared values	---
IRD	---	Device number for the input file	---
IRVRS	---	Flag for reversing the shot line direction; $= 0$: normal view $= 1$: reversed view	---
ITAB(I) $I \leq 500$	---	Table look-up code for the Ith component	---
ITC	---	Dimension of the component arrays	---
IU(I) $I \leq 500$	---	Multiply vulnerable flag for Ith component; $= 1$: multiply vulnerable $= 0$: singly vulnerable	---
IWR	---	Device number for the output file	---
IY(I) $I \leq 500$	---	System number for the Ith component	---
K	---	Number of comparisons needed to sort the sublists by insertion in Subroutine FSORT	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program PRERD (Concluded)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
M	---	Increment size for each sublist, the number of sublists in Subroutine FSORT	---
MAT(I) $I \leq 500$	---	Material code number for the Ith component	---
NOCOMP	---	Number of components in the target model	---
NOPNTS(I) $I \leq 500$	---	Number of Pk values for Ith component	---
NTMAX	---	Number of time intervals in array TIMAX, number of time intervals for shot line breakdown	---
PKNBR	---	Maximum number of penetration depths and Pk values that can be specified for each component; the first dimension of the DEPTH and PKVAL arrays	---
PKVAL(J,I) $J \leq 10$, $I \leq 500$	Pk_N	Pk associated with penetration to a specified depth for the Ith component	---
RHOF(I) $I \leq 500$	---	Ith component density factor	---
THINFL(I) $I \leq 500$	---	Ith component wall thickness for tubes modeled in the influence mode	meters
TIMAX(I) $I \leq 25$	---	Upper boundary for Ith time interval, for shot line breakdown by times needed to reach the maximum shot line Pk	seconds
TINIT(I) $I \leq 500$	---	Initial operating temperature for the Ith component	^oC

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (including subroutines BSRCH, FSORT, PROPY, RAT, RDATA, REVRSE, and TABLE)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
ALP	α	Component coupling coefficient, absorption factor	---
ALPC	---	Component coupling coefficient computed as a product of ALPF and ALPT	---
ALPF	---	Intensity dependent factor of the coupling coefficient	---
ALPHA(I,J) $I<11$, $J\leq16$	---	Intensity dependent factors of the coefficient for the Ith material code and the Jth flux argument	---
ALPHAT(I,J) $I\leq11$, $J\leq9$	---	Coupling coefficient correction factors for material thickness for the Ith material code and the Jth thickness argument	---
ALPT	---	Thickness dependent correction factor of the coupling coefficient	---
AVAL	---	One of the factors used to compute the arguments in the coupling coefficient interpolation schemes	---
AZ	---	Attack azimuth angle	degrees
CINCH	---	Conversion factor to convert input length units to inches; ≤ 0.0: set conversion factor to 1.0 (units are in inches) > 0.0: use conversion factor (units are not in inches)	inches/unit length

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
CMETER	---	Conversion factor to convert input length units to meters	meter/unit length
CONDUC(I,1,K) $I \leq 11$, $K \leq 10$	---	Temperature arguments for the thermal conductivity of the Ith material type; Array element (I,1,1) is the number of arguments in positions K = 2, 3, . . .	^{o}C
CONDUC(I,2,K) $I \leq 11$, $K \leq 10$	---	Thermal conductivity for the Ith material code at the Kth temperature argument	watts/m ^{o}C
CP	$CP\ell$	Specific heat for a liquified material with a temperature $\frac{T^* + T_m}{2}$	joules/kgm ^{o}C
CPS	Cps	Solid component specific heat at the average temperature $\frac{T_m + T_1}{2}$	joules/kgm ^{o}C
CVAL	---	Thermal conductivity interpolated from the CONDUC array, a function of temperature and material type	watts/m ^{o}C
DENSIT(I) $I \leq 11$	---	Component density for the Ith material code	kgm/m^3
DEPTH(J,I) $J \leq 10$, $I \leq 500$	---	Penetration depth of the Ith component to achieve a Pk = PKVAL(J,I)	inches
DISLFT	---	Distance not yet penetrated in an encounter	meters
DP	Z	Component effective thickness	meters

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
DUM	---	Dummy variable used to read a value from the LOS file which will not be used	---
EL	---	Attack elevation angle	degrees
EM	---	Slope of one of the coupling coefficient functions being interpolated in Subroutine PROPY	---
FB	---	Weapon flux at the beginning of a time interval	watts/cm^2
FE	---	Weapon flux at the end of a time interval	watts/cm^2
FLX	F	Flux, weapon intensity, rate of energy reception per unit of area	watts/cm^2
FTIM(I) $I \leq 25$	---	Time arguments for the flux distribution table	seconds
FXCM(I) $I \leq 25$	---	Weapon flux at time FTIM(I)	watts/cm^2
FXM(I) $I \leq 25$	---	Weapon flux at time FTIM(I)	watts/m^2
GRID	---	Grid size of one grid square in the viewing plane	varies
HEATFU(I) $I \leq 11$	---	Heat of fusion for a component with the Ith material code	joules/kgm
I1(I) $I \leq 200$	---	Number of encounters on a shot line for each new shot line; otherwise, the component number in the encounter with penetration times PKTIME(J,I) and PLS(I) (equals 9999 to indicate end of view)	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
IANA(I) $I \leq 500$	---	Analysis type of the Ith component; = 1: melt in place = 2: melt removed	---
IBEG	---	Beginning element in the interval being searched in Subroutine BSRCH	---
ICODE	---	End of shot line indicator in LOS data file; = 9: last encounter on the shot line ≠ 9: not last shot line encounter	---
ICOMP(I) $I \leq 500$	---	Component identification number for the Ith component	---
IDVEH	---	Unused variable on first record of LOS file	---
IECHO	---	Echo flag for QKPK time file; = 0: penetration times not printed on device IWR = 1: penetration times also printed on device IWR	---
IEND	---	End element in the interval being searched in Subroutine BSRCH	---
IFG(I) $I \leq 500$	---	Criticality flag for the Ith component; = 0: noncritical component ≠ 0: critical component	---
IFILL	---	Subscript for the output arrays, counts the number of array elements which are "full" with data not yet written on the output file	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
IFLAG	---	Subscript in the LOC array corresponding to the last critical encounter number on a shot line; = 0: shot line has no critical encounters	---
IFM	---	One less than the number of points in the flux distribution	---
IFMAX	---	Number of points in the flux distribution table	---
IFX	---	Maximum number of points allowed in the flux distribution, the array dimension (=25)	---
IFXM	---	Subscript for the next to last element in the MAXT array (=26)	---
IFXX	---	Subscript for the last element in the MAXT array (=27)	---
IIN	---	Device number for the LOS file (=1)	---
IM	---	Larger of two subscripts of compared values in Subroutine FSORT	---
IOUT	---	Device number for the binary penetration times output file (=2)	---
IR	---	Index pointing to the current flux level in the flux distribution table	---
IRD	---	Device number for the QKPK formatted input file (=5)	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
IRNOW	---	Index pointing to the current flux level while executing the encounter loop	---
IRUSED	---	Index pointing to last flux level used while executing the encounter loop	---
IRVRS	---	Reverse shot line flag; = 0: normal shot line processing; = 1: process shot line with encounters in reverse order	
ITAB(I) $I \leq 500$	---	Table look-up code for the Ith component; = 1: plexiglass; = 2: triplex dielectric; = 3: fibrite; = 4: rubber; = 5: glass; = 6: fiber glass; = 7: pyroceram; = 8: glass fiber epoxy; = 9: graphite epoxy	---
ITC	---	Dimension of the component information arrays (=500)	---
ITER	---	= 0: compute all encounter properties in Subroutine PROPY; = 1: compute only the temperature dependent properties in Subroutine PROPY	---
IU(I) $I \leq 500$	---	Multiply vulnerable flag for the Ith component; = 0: singly vulnerable; = 1: multiply vulnerable	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
IWR	---	Device number for the QKPK formatted output file (=6)	---
IY(I) $I \leq 500$	---	System number for the Ith component	---
J	---	Subscript for the component information arrays corresponding to the encountered correspondent	---
JCOMP	---	Encountered component identification number	---
JF	---	Subscript for flux level at the end of a time interval	---
JFF	---	Subscript for flux level at the end of the previous time interval	---
JH(1,J) $J \leq 170$	---	End of shot line flag in the LOS input file; $= 9$: end of shot line $\neq 9$: not end of shot line	---
JH(2,J) $I \leq 170$	---	Encountered component number or when = 0, the end of view flag for the LOS data file	---
JH(3,J) $J \leq 170$	---	Component LOS thickness multiplied by 100	inches/100
JH(4,J) $I \leq 170$	---	1000.0 times the secant of the entrance obliquity angle	---
JH(5,J) $I \leq 170$	---	1000.0 times the secant of the exit obliquity angle (When the LOS file is assembled by using Program CONMAG, JH(4,J) = JH(5,J))	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
JHT(I) $I \leq 100$	---	Component identification number for the Ith encounter on a shot line	---
K	---	In Subroutine FSORT, the number of comparisons needed to sort M sublists by insertion	---
LOC(I) $I \leq 100$	---	Subscript in the component information arrays for the component in the Ith shot line encounter	---
M	---	Increment size for each sublist, the number of sublists in Subroutine FSORT	---
MAT(I) $I \leq 500$	---	Material code for the Ith component; = 1: 2024 aluminum painted surface; = 2: 7075 aluminum painted surface; = 3: 5456 aluminum painted surface; = 4: 6AL4V titanium painted surface; = 5: pure titanium painted surface; = 6: VM65-1 magnesium alloy painted surface; = 7: ML5 magnesium alloy painted surface; = 8: AZ31B magnesium alloy painted surface; = 9: EI435 nickel-chromium alloy painted surface; = 10: 304 stainless steel painted surface; = 11: copper bare surface	---
MATL	---	Material code for the encountered component	---

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
MAXT(I) I≤25	---	Number of shot lines that reach their maximum Pk during the Ith time interval	---
MAXT(26)	---	The number of shot lines that reach their maximum Pk after the last time interval	---
MAXT(27)	---	Number of noncritical shot lines, number of shot lines with zero critical encounters	
MVAL	---	Subscript of the last temperature argument in the CONDUC or SPECIF arrays	---
N	---	Subscript of the component information arrays corresponding to the encountered component in Subroutine PROPY (equivalent to J in Subroutine RAT)	---
NCRIT	---	Number of encounters with critical components on a shot line	---
NENC	---	Number of encounters on a shot line	---
NOPNTS(I) I≤500	---	Number of Pk values for Ith component	---
NSL	---	Number of shot lines	---
NTMAX	---	Number of time intervals for breakdown of the shot line by the times required to reach maximum shot line Pk	---
OBQ(I) I≤100	---	Secant of the entrance obliquity angle for the Ith shot line encounter	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
PEAKF	---	Weapon intensity, flux	watts/cm^2
PKNBR	---	Maximum number of penetration depths and Pk values that can be specified for each component; the first dimension of the DEPTH and PKVAL arrays	---
PKTIME(1,I) $I \leq 200$	---	y-coordinate in the viewing plane for each new shot line; otherwise, the time needed to penetrate from the start of the shot line to the depth DEPTH(1,I) in the encounter with component number I1(I)	inches or seconds
PKTIME(2,I) $I \leq 200$	---	z-coordinate in the viewing plane for each new shot line; otherwise, the time needed to penetrate from the start of the shot line to the depth DEPTH(2,I) in the encounter with component number I1(I)	inches or seconds
PKTIME(J,I) $J \leq 10$, $I \leq 200$	---	Time needed to penetrate from the start of the shot line to the depth DEPTH(J,I) in the encounter with component number I1(I)	seconds
PKVAL(J,I) $J \leq 10$, $I \leq 200$	Pk_N	Pk associated with penetration to a specified depth for the Ith component	---
PLS(I) $I \leq 200$	---	Equals zero for each new shot line; otherwise, the time needed to penetrate from the start of the shot line through the LOS thickness of component number I1(I)	--- or seconds
RADEXP	---	Unused variable, read on first record of the LOS file	---

LIST OF ABBREVIATIONS AND SYMBOLS
(SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
RATE	R	Penetration rate of a Directed High Energy Weapon against a component	m/sec
RATES(I) $I \leq 100$	---	Penetration rate for the Ith shot line encounter in Subroutine RAT (equivalent to RATES(I,J) for Jth flux level in Program QKPK)	m/sec
RATES(I,J) $I \leq 100$, $J \leq 25$	---	Penetration rate for the Ith shot line encounter at the Jth level in the flux distribution	m/sec
RBAR	---	Average penetration rate for complete perforation of a specific component	m/sec
RHO	ρ	Component material density	kgm/m^3
RHOF(I) $I \leq 500$	---	Density factor ($0 \leq RHOF \leq 1$) for the Ith component; used to compute the actual component LOS thickness for components not modeled in the influence mode	---
RHX	---	Component material density	kgm/m^3
SH(1,J) $J \leq 170$	---	Shot line y-coordinate in the viewing plane	varies
SH(2,J) $J \leq 170$	---	Shot line z-coordinate in the viewing plane	varies
SPECIF(I,1,K) $I \leq 11$, $2 \leq K \leq 10$	---	Specific heat for the Ith material code at the Kth temperature argument	joules/ kgm ^oC
STAR	---	Maximum energy absorption rate for an encounter in which the surface temperature exceeds the vapor temperature	$watts/m^2$

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
SVAL	---	Specific heat for an encounter in Subroutine PROPY (equivalent to CP in Subroutine RAT)	joules/kgm ^{o}C
T	---	Temperature of a component	^{o}C
TB	---	Beginning time of a time interval	seconds
TE	---	Ending time of a time interval	seconds
TEMP	---	Component temperature	^{o}C
THINFL(I) $I \leq 500$	---	Wall thickness for the Ith component modeled in the influence mode	varies
THRAT1(I) $I \leq 500$	---	Maximum of the ratios (DEPTH(1,I)*OBQ/LOS) for all encounters of the Ith component	---
THRAT2(I) $I \leq 500$	---	Maximum of the ratios (DEPTH(NOPNTS(I),I)*OBQ/LOS) for all encounters of the Ith component	---
TIMAX(I) $I \leq 25$	---	Upper boundary on the Ith time interval for breakdown of the shot lines by times needed to reach maximum shot line Pk; lower boundary of the Ith interval is TIMAX(I-1) or 0.0 for the first interval	seconds
TINIT(I) $I \leq 500$	T_1	Initial operating temperature for the Ith component	^{o}C
TLOS	---	Time for complete perforation of a component from the end of the previous encounter	seconds

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Continued)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
TMAX	---	Minimum time required to complete all encounters on a shot line	seconds
TMAX1	---	Earliest time that the maximum shot line Pk will occur	seconds
TMELT(I,1) $I \leq 11$	---	Melting temperature of a component with the Ith material code	^{o}C
TMELT(I,2) $I \leq 11$	---	Vapor temperature of a component with the Ith material code	^{o}C
TMLT	T_m	Component melting temperature	^{o}C
TNOW	---	Time at which a component is perforated after the beginning of the shot line	seconds
TSTAR	T*	Component outer surface temperature	^{o}C
TUSED	---	Time used from the start of the shot line to perforate a component	seconds
TVAP	T_V	Component vaporization temperature	^{o}C
XK	k	Thermal conductivity of the component	watts/m ^{o}C
XLAMBD	λ	Heat of fusion for a component, the energy used to change from the solid to the liquid state	joules/kgm
XLOS(I) $I \leq 100$	---	Actual component LOS thickness for the Ith shot line encounter	inches or meters

LIST OF ABBREVIATIONS AND SYMBOLS (SIMULATION MODEL)

Program QKPK (Concluded)

Abbreviation or Symbol	Equivalent in Mathematical Model	Definition	Units
XVAL	---	Argument increment size in the coupling coefficient interpolation routines of Subroutine PROPY	watts/m^2 or meters
YMAX	---	Maximum y-coordinate for all shot lines in the viewing plane	varies
YMIN	---	Minimum y-coordinate for all shot lines in the viewing plane	varies
YVAL	---	Function increment size in the coupling coefficient interpolation routines in Subroutine PROPY	---
ZMAX	---	Maximum z-coordinate of all shot lines in the viewing plane	varies
ZMIN	---	Minimum z-coordinate of all shot lines in the viewing plane	varies

Encounter Descriptions

Card: C4

A_1	B_1	C_1	D_1	E_1	A_2	B_2	C_2	D_2	E_2	A_3	B_3	C_3	D_3	E_3	╳

ID	Parameter	Units	Format	Columns	Description
A_1	ICM(2)	---	I4	1-4	Intersected component number.
B_1	TM(2)	inches	F7.2	5-11	Component line-of-sight thickness.
C_1	QM(2)	degrees	F5.1	12-16	Entrance obliquity angle.
D_1	ISM(2)	---	I3	17-19	Space code for space following component (=9 for last encounter for given shot line).
E_1	TL(2)	inches	F7.2	20-26	Line-of-sight distance through space following component.
.	.		.	.	. } second encounter
.	.		.	.	.
.	.		.	.	.
A_3	ICM(4)	---	I4	53-56	. } third encounter
B_3	TM(4)	inches	F7.2	57-63	.
C_3	QM(4)	degrees	F5.1	64-68	.
D_3	ISM(4)	---	I3	69-71	
E_3	TL(4)	inches	F7.2	72-78	

Card: C4

PROGRAM PEAKAY INPUT

Program PEAKAY requires two input files. A short formatted file with the time interval at which vulnerable areas are to be computed is read from Logical Unit Number 5. A binary penetration times file, which is the output of Program QKPK, is read from Logical Unit Number 2. The user must assemble the formatted input file.

PEAKAY Formatted Data Deck

This deck contains two cards; the first contains the number of time intervals and a component vulnerability area flag. The second contains the upper boundaries of the time intervals. There must be no more than 10 time intervals. The vulnerability area flag determines whether the vulnerable areas computed for the components are true values (flag = 1) or incremental values (flag = 0). The first time interval is assumed to start at time 0.00. All other time intervals begin when the preceding interval ends. The times on the second card must be in ascending order. Figure 5-2 shows the PEAKAY data deck setup using cards PK1 and PK2.

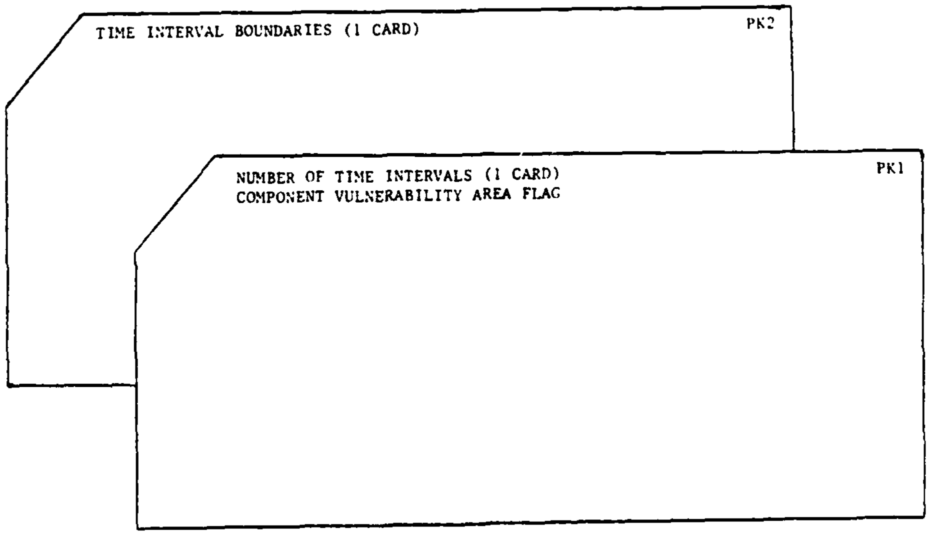

TIME INTERVAL BOUNDARIES (1 CARD)
PK2
NUMBER OF TIME INTERVALS (1 CARD)
COMPONENT VULNERABILITY AREA FLAG
PK1

FIGURE 5-2. PEAKAY Data Deck Setup.

Number of Time Intervals					Card: PK1
ID	Parameter	Units	Format	Columns	Description
A	NTIME	---	I5	1-5	The number of time intervals for which kill probabilities and vulnerable areas are to be computed. NTIME must be less than or equal to 10.
B	AVFLAG	---	I5	6-10	Flag indicating whether the component vulnerable areas are to be calculated as incremental or true 0 = incremental values 1 = true values

Time Interval Boundaries

Card: PK2

	A	B	C	D	E	F	G	H	I	J

ID	Parameter	Units	Format	Columns	Description
A	TIMES(1)	seconds	F8.2	1-8	Upper boundary of the first time interval for which Pk's and vulnerable areas are to be computed.
B	TIMES(2)	seconds	F8.2	9-16	Upper boundary of the second time interval for which Pk's and vulnerable areas are to be computed.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	TIMES (NTIME)	seconds	F8.2		Upper boundary of the last time interval for which Pk's and vulnerable areas are to be computed. Note: The lower boundary for the first interval is 0.00. The lower boundary for all other intervals equals the upper boundary of the preceding interval. NTIME $\leq$ 10.

Card: PK2

PEAKAY Binary Input File

The penetration times for each encounter along each shot line are read from the PEAKAY binary input file. These data are the output of Program QKPK. The records vary in length and the total number of records depends on the amount of data. There are three types of records on this file.

Viewing Plane Description

Figure 5-3 shows the format for the first record of the binary file which contains nine words. The data on this record include the attack aspect angles, grid size, minimum and maximum shot line coordinates, and the number of components in the target model.

Component Data and Flux Distribution

This is the second record on the file and it contains four arrays of component information, the shot line reverse flag, the number of flux distribution points, and the time and flux values from the flux distribution. It has 4*NOCOMP + 2 + 2*IFMAX words. Figure 5-4 lists the parameters and definitions on this record.

Penetration Times

Figure 5-5 shows the format for the third and all subsequent records on this file. It contains three arrays with 2400 total words per record. The second subscript of the first array corresponds to the subscripts of the other two arrays. They define either a shot line or an encounter along a shot line. When the value of an element of the fourth array, I1, equals 9999, the end of all data for the view is indicated and the rest of the data on the binary input file is not processed.

Record Number	1	2	3	...	Last
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 1

Word	Parameter	Units	Definition
1	AZ	degrees	Attack azimuth angle
2	EL	degrees	Attack elevation angle
3	GRID	inches	Grid cell size
4	IDVEH	---	Currently not used
5	YMAX	inches	Maximum y-coordinate for all shot lines
6	YMIN	inches	Minimum y-coordinate for all shot lines
7	ZMAX	inches	Maximum z-coordinate for all shot lines
8	ZMIN	inches	Minimum z-coordinate for all shot lines
9	NOCOMP	---	Number of components in the target model

FIGURE 5-3. PEAKAY Binary Input, Viewing Plane Description.

Record Number	1	2	3	...	Last
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 2			
Word	**Parameter**	**Units**	**Definition**
1	ICOMP(1)	---	Component identification number for first component
2	IFG(1)	---	Criticality flag for the first component: =0: noncritical; ≠0: critical
3	IY(1)	---	System number for the first component
4	IU(1)	---	Vulnerability flag for first component: =0: singly vulnerable; =1: multiply vulnerable
5	ICOMP(2)	---	Component identification number for second component
.	.		
.	.		
.	.		
.	ICOMP(NOCOMP)	---	Component identification number for NOCOMP*th* component
.	IFG(NOCOMP)	---	Criticality flag for the NOCOMP*th* component
.	IY(NOCOMP)	---	System number for the NOCOMP*th* component

FIGURE 5-4. PEAKAY Binary Input, Component Data and Flux Distribution (Page 1 of 2).

Record Number	1	2	3	...	Last
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 2			
Word	**Parameter**	**Units**	**Definition**
4*NOCOMP	IU(NOCOMP)	---	Vulnerability flag for NOCOMPth component
.	IRVRS	---	Shot line reverse flag: =0: normal encounter order =1: reversed encounter order
4*NOCOMP+2	IFMAX	---	Number of points in the flux distribution table
.	FTIM(1)	seconds	First time argument in the flux distribution
.	FXCM(1)	watts/cm^2	Weapon intensity, flux at time FTIM(1)
.	FTIM(2)	seconds	Second time argument in the flux distribution
.	FXCM(2)	watts/cm^2	Weapon intensity, flux at time FTIM(2)
	. . .		
	FTIM(IFMAX)	seconds	Last time argument in the flux distribution
4*NOCOMP +2+2*IFMAX	FXCM(IFMAX)	watts/cm	Weapon intensity, flux at time FTIM(IFMAX)

FIGURE 5-4. PEAKAY Binary Input, Component Data and Flux Distribution (Page 2 of 2).

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 through Last			
Word	Parameter	Units	Definition
1	R1(1,1)	inches	y-coordinate of a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(1) in an encounter with component I1(1)
2	R1(2,1)	inches	z-coordinate of a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(2) in an encounter with component I1(1)
3	R1(3,1)	seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(3) in and encounter with component I1(1)
. . .	. . .		. . .
10	R1(10,1)	seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(10) in an encounter with component I1(1)

FIGURE 5-5. PEAKAY Binary Input, Penetration Times (page 1 of 4).

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 through Last			
Word	**Parameter**	**Units**	**Definition**
11	R1(1,2)	inches	y-coordinate of a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(1) in an encounter with component I1(2)
12	R1(2,2)	inches	z-coordinate of a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(2) in an encounter with component I1(2)
⋮	⋮		⋮
20	R1(10,2)	seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(10) in an encounter with component I1(2)
⋮	⋮		⋮
2000	R1(10,200)	seconds	Time needed to penetrate from the start of the shot line to normal depth DEPTH(10) in an encounter with component I1(200)

FIGURE 5-5. PEAKAY Binary Input, Penetration Times (page 2 of 4).

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 through Last			
Word	Parameter	Units	Definition
2001	R3(1)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(1)
2002	R3(2)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(2)
. . .	. . .		. . .
2200	R3(200)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(200)

FIGURE 5-5. PEAKAY Binary Input, Penetration Times (Page 3 of 4).

Record Number	1	2	3	. . .	LAST
Number	9	4*NOCOMP + 2	2400		2400

Record Number 3 through Last			
Word	**Parameter**	**Units**	**Definition**
2201	I1(1)	---	Number of encounters on a new shot line
		or	or
		---	Component in the encounter with penetration times R1(1,1), R1(2,1), . . ., R1(10,1), and R3(1) (the subscript in the ICOMP array for the corresponding component identification number)
		or	or
		---	Equals 9999 to indicate the end of view
. . .	. . .		. . .
2400	I1(200)	---	Number of encounters on a new shot line
		or	or
		---	Component in the encounter with penetration times R1(1,200), R1(2,200), . . ., R1(10,200), and R3(200) (the subscript in the ICOMP array for the corresponding component identification number)
		or	
		---	Equals 9999 to indicate the end of view

FIGURE 5-5. PEAKAY Binary Input, Penetration Times (Page 4 of 4).

PROGRAM PRERD INPUT

One formatted deck is the only input necessary for Program PRERD. It is read from Logical Unit Number 5 during execution of Subroutine READIN. The data on these cards must be assembled by the user and are very important in order to obtain good results from QKLOOK. The importance and complexity of these data are the reasons that Program PRERD must be used. It will flag possible errors in this data deck so that the user can correct errors before executing Program QKPK which reads the same deck. Figure 5-6 shows the data deck setup with the eight types of cards in this deck.

Number of Components

This is the first card in the deck and contains the number of components in the target model. The array dimensions currently allow a maximum of 498 components.

Reverse Flag and Conversion Factor

This is the second card in the deck and contains the shot line reverse flag and a length units conversion factor. The reverse flag enables a user to obtain data from the view used by the shot line generating program or the opposite view without a second execution of the shot line generating program. When the shot lines are being reversed using data from Program MAGIC, the new entrance obliquity angles equal the old entrance obliquity angles, because Program MAGIC output does not include exit obliquity angles. The length conversion factor allows the input lengths to be in any one unit of measure. The data card description lists the input variables from both files which are converted using CINCH.

Number of Flux Distribution Points

The third data card contains the number of points in the flux distribution table which is on card types QK4 and QK5. The current program allows a maximum of 25 flux distribution points.

Flux Levels

This card type is fourth in the data deck and contains the weapon flux levels at each point in the flux distribution. Flux levels must be greater than 0.00 and less than or equal to 60000.0 watts/cm^2. There will be one, two, or three of these cards consecutively depending on the number of flux points, IFMAX.

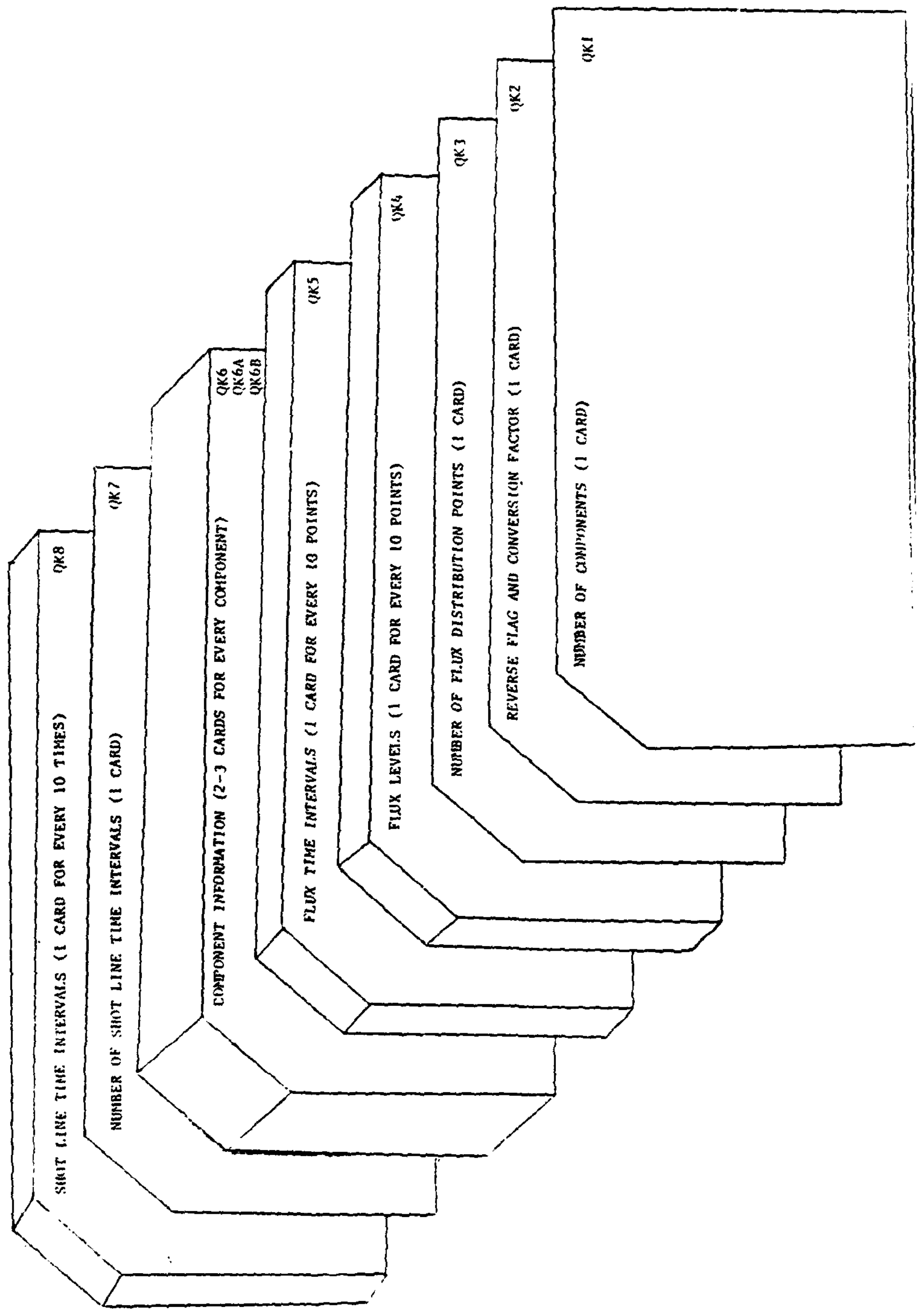

QK1
NUMBER OF COMPONENTS (1 CARD)
QK2
REVERSE FLAG AND CONVERSION FACTOR (1 CARD)
QK3
NUMBER OF FLUX DISTRIBUTION POINTS (1 CARD)
QK4
FLUX LEVELS (1 CARD FOR EVERY 10 POINTS)
QK5
FLUX TIME INTERVALS (1 CARD FOR EVERY 10 POINTS)
QK6
QK6A
QK6B
COMPONENT INFORMATION (2-3 CARDS FOR EVERY COMPONENT)
QK7
NUMBER OF SHOT LINE TIME INTERVALS (1 CARD)
QK8
SHOT LINE TIME INTERVALS (1 CARD FOR EVERY 10 TIMES)

FIGURE 5-6. Program PRERD Data Deck Setup.

Flux Time Intervals

One, two, or three of these cards, depending on IFMAX, follow the flux levels in the data deck. The times in the flux distribution are listed on these cards. These times correspond by subscript with the flux levels on card type QK4.

Component Information

Cards QK6 and QK6A are included for every component in the target model, NOCOMP. QK6B is included for a component when more than four penetration depths and associated Pk's are to be entered. The component identification numbers must correspond to component numbers used in the target model for the shot line generating program. The order of the components in the deck can vary, because Subroutine FSORT is used to sort these data into ascending order by component number.

Number of Shot Line Time Intervals

Card QK7 contains the number of time intervals for shot line breakdown by times required to reach maximum shot line Pk. This and the last card type (QK7 and QK8) are optional input. If cards QK7 and QK8 are not included, the values from cards QK3 and QK5 will be used as default data. If card QK7 is included, then card QK8 is required.

Shot Line Time Intervals

This card type contains the time intervals for shot line breakdown by times required to reach maximum shot line Pk. One time is listed for each time interval. The beginning time for each interval is assumed to equal the ending time of the preceding interval. The first time interval begins at time 0.00. There are one, two, or three of these cards depending on the number of intervals, NTMAX. Card type QK8 may be excluded from the deck if card QK7 is also excluded. In this case, the times on card QK5 will be used as default data.

Number of Components — Card: QK1

ID	Parameter	Units	Format	Columns	Description
A	NOCOMP	---	I5	1-5	Number of components in the target model. $1 \leq \text{NOCOMP} \leq 498$.

Card: QK1

Component Information | Card: QK6

A B C D E F G H I J K L

ID	Parameter	Units	Format	Columns	Description
A	ICOMP(I)	---	I5	1-5	Component identification number for Ith component
B	MAT(I)	---	I3	6-8	Material code number for Ith component: 0 = use table look-up 1 = 2024AL painted 2 = 7075AL painted 3 = 5456AL painted 4 = 6AL4V TI 5 = pure TI painted 6 = VM 65-1 MAG alloy painted 7 = ML5 MAG alloy painted 8 = AZ31B MAG alloy painted 9 = EI435 nickel chromium painted 10 = 304 stainless steel painted 11 = CU bare surface
C	IFG(I)	---	I1	10	Criticality flag for Ith component: 0 = noncritical 1 = critical 2 = critical 3 = critical
D	ITAB(I)	---	I2	11-12	Table look-up code: 0 = use material code 1 = plexiglass 2 = triplex Dielectric 3 = fibrite 4 = rubber 5 = glass 6 = fiber glass 7 = pyroceram 8 = glass fiber epoxy 9 = graphite epoxy
E	IANA(I)	---	I1	26	Analysis type: 1 = melt-in-place 2 = melt-removed

Card: QK6

Component Information (Continued) — Card: QK6 (concld.)

ID	Parameter	Units	Format	Columns	Description
F	TINIT(I)	^{o}C	F5.0	27-31	Initial operating temperature. $0.0 < TINIT$
G	THINFL(I)	*	F3.2	64-66	Wall thickness for influence mode component. $0 \leq THINFL$.
H	RHOF(I)		F3.2	74-76	Density factor $0.0 \leq RHOF \leq 1.0$.
I	IY(I)	---	I2	77-78	System number $0 \leq IY \leq 10$.
J	IU(I)	---	I2	79-80	Multiply-vulnerable component flag: 0 = singly vulnerable; 1 = multiply vulnerable. Note: This card is repeated NOCOMP times.

Card: QK6 (concld.)

*Any one length unit may be used for these. They are converte! by using the factor CINCH.

Component Information (Continued) — Card: QK6A

ID	Parameter	Units	Format	Columns	Description
A	NOPNTS(I)	---	I5	1-5	Number of points for which Pk values are specified. $2 \le \text{NOPNTS} \le 10$.
B	DEPTH(1,I)	*	F7.4	6-12	Minimum penetration depth normal to the surface for Pk = PKVAL(1,I) $0.0 \le \text{DEPTH}(1,I)$
C	PKVAL(1,I)	---	F7.4	13-19	Pk of component I when component penetrated to a depth of DEPTH(1,I) $0.0 \le \text{PKVAL}(1,I) \le 1.0$.
D	DEPTH(2,I)	*	F7.4	20-26	Minimum penetration depth normal to the surface for Pk = PKVAL(2,I) $\text{DEPTH}(1,I) \le \text{DEPTH}(2,I)$
E	PKVAL(2,I)	---	F7.4	27-33	Pk of component I when component penetrated to a depth of DEPTH(2,I) $\text{PKVAL}(1,I) \le \text{PKVAL}(2,I) \le 1.0$.
. . .	. . .		. . .	. . .	
J	DEPTH(5,I)	*	F7.4	62-68	Minimum penetration depth normal to the surface for Pk = PKVAL(5,I) $\text{DEPTH}(4,I) \le \text{DEPTH}(5,I)$

Card: QK6A

*Any one length unit may be used for these. They are converted by using the factor CINCH.

Component Information (Concluded) — Card: QK6B

ID	Parameter	Units	Format	Columns	Description
A	PKVAL(5,I)	---	F7.4	1-7	Pk of component I when component penetrated to a depth of DEPTH(5,I) PKVAL(4,I) $\leq$ PKVAL(5,I) $\leq$ 1.0.
B	DEPTH(6,I)	*	F7.4	8-14	Minimum penetration depth normal to the surface for Pk = PKVAL(6,I) DEPTH(5,I) $\leq$ DEPTH(6,I)
⋮	⋮		⋮	⋮	
J	DEPTH(10,I)	*	F7.4	64-70	Minimum penetration depth normal to the surface for Pk = PKVAL(10,I) DEPTH(9,I) $\leq$ DEPTH(10,I)
K	PKVAL(10,I)	---	F7.4	71-78	Pk of component I when component penetrated to a depth of DEPTH(10,I) PKVAL(9,I) $\leq$ PKVAL(10,I)

Card: QK6B

*Any one length unit may be used for these. They are converted by using the factor CINCH.

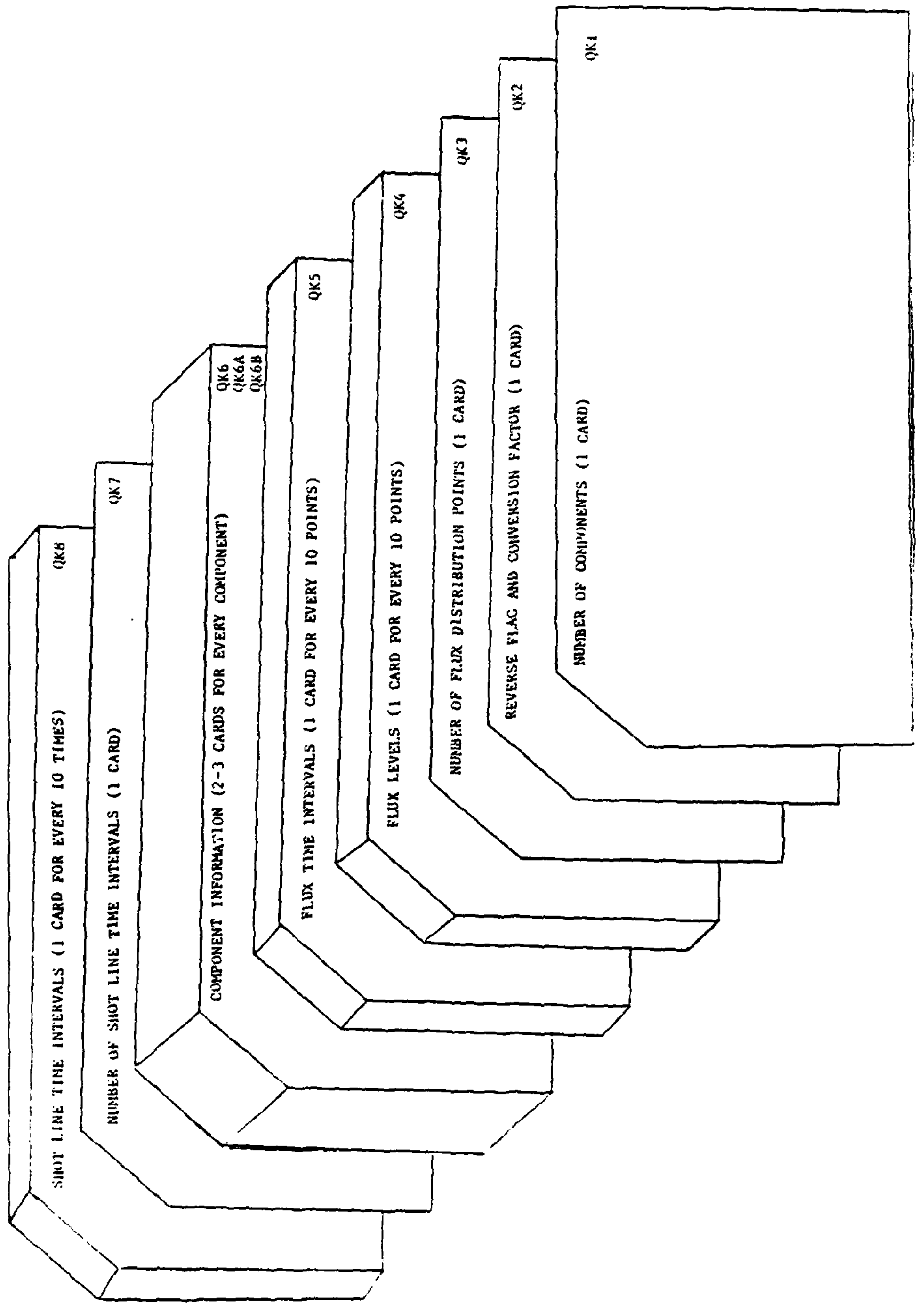

QK1
NUMBER OF COMPONENTS (1 CARD)
QK2
REVERSE FLAG AND CONVERSION FACTOR (1 CARD)
QK3
NUMBER OF FLUX DISTRIBUTION POINTS (1 CARD)
QK4
FLUX LEVELS (1 CARD FOR EVERY 10 POINTS)
QK5
FLUX TIME INTERVALS (1 CARD FOR EVERY 10 POINTS)
QK6
QK6A
QK6B
COMPONENT INFORMATION (2-3 CARDS FOR EVERY COMPONENT)
QK7
NUMBER OF SHOT LINE TIME INTERVALS (1 CARD)
QK8
SHOT LINE TIME INTERVALS (1 CARD FOR EVERY 10 TIMES)

FIGURE 5-7. Program QKPK Data Deck Setup.

QKPK Binary Input File

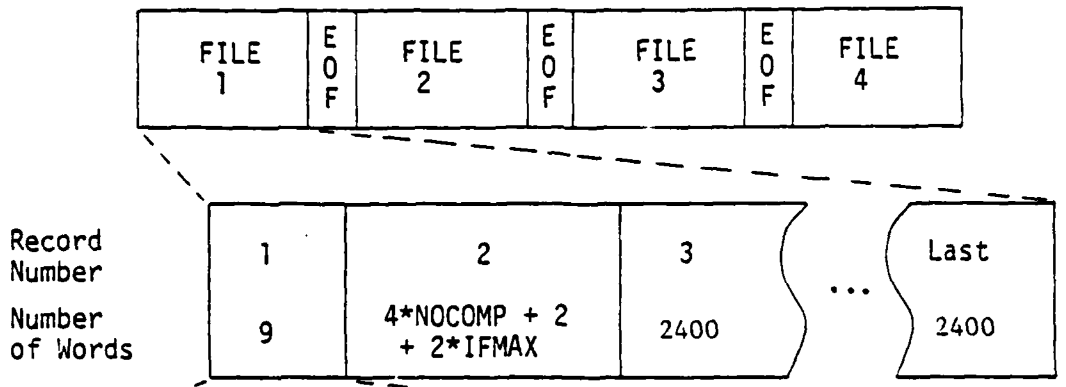
This file contains encounter information for each encounter along each shot line from one viewing plane. It may be the output from Program CONMAG (converts MAGIC to QKLOOK format), FASTGEN, or possibly another shot line generating program. The order of data on this file must match the order described in Figures 5-8 and 5-9 in order to execute Program QKPK. Some shot line generating programs may produce data for more than one view. Program QKPK halts after processing one view, so this discussion will assume the record containing the first end-of-view flag, is the end of the input file. The units for some of the parameters in this file are described as "varies" on the record description forms. These are the parameters which are converted using CINCH, the conversion factor from card QK2 in the QKPK formatted data deck.

Viewing Plane Description

This is always the first record for a view and is described in Figure 5-8. It contains nine parameters which define the attack angles, grid size, and minimum and maximum shot line coordinates for all shot lines in the viewing plane.

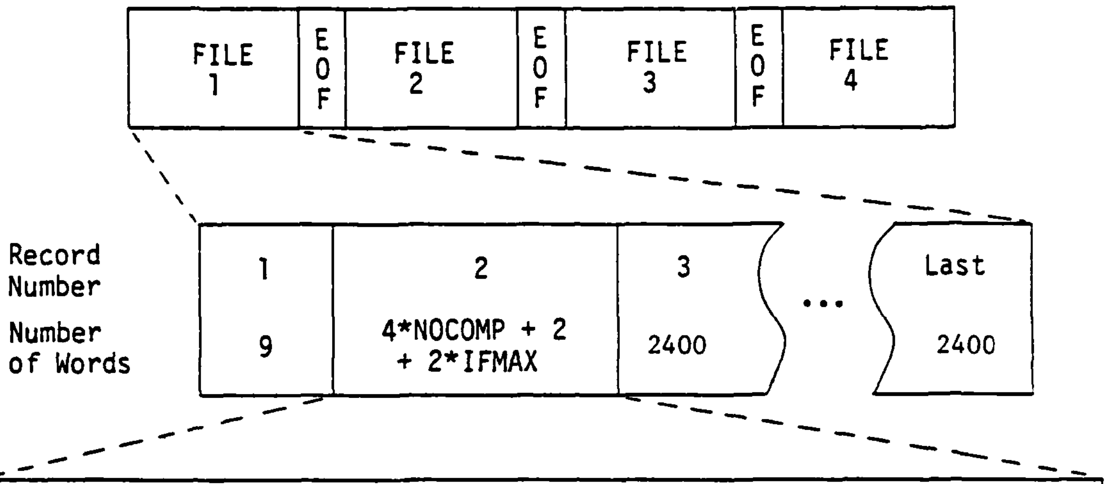
Encounter Data

Figure 5-9 depicts the data arrangement for the second and all subsequent records on this binary file. These data describe each encounter along each shot line. The shot line coordinates are duplicated when a shot line has more than one encounter. The last encounter for a shot line is indicated when JH(1,J) equals 9. If the encountered component number JH(2,J) equals 0, this marks the end of all data for the view, and all subsequent values on the record are ignored. The record containing the end-of-view flag is also the last record read from the binary input file.



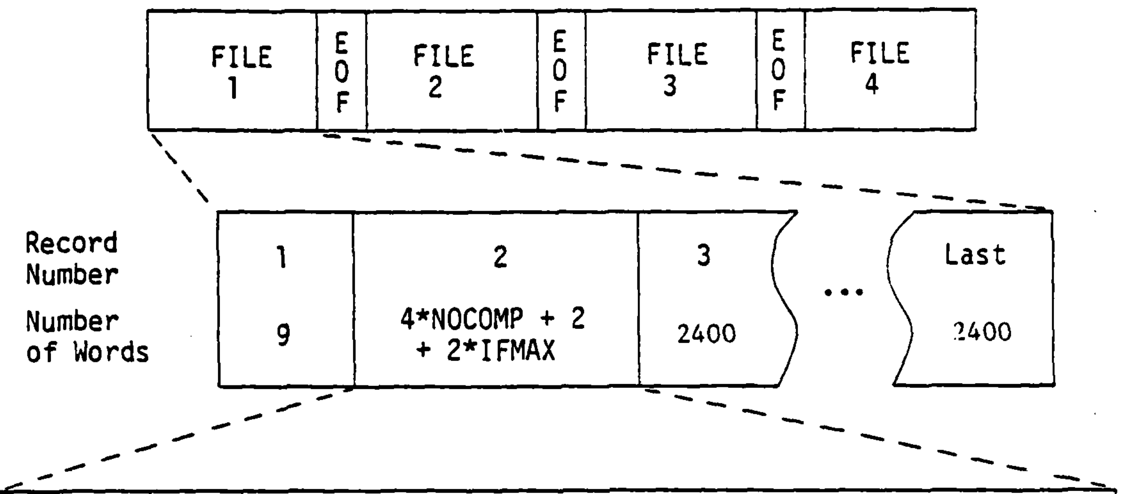
Record Number 1			
Word	Parameter	Units	Definition
1	AZ	degrees	Attack azimuth angle
2	EL	degrees	Attack elevation angle
3	GRID	inches	Grid cell size
4	IDVEH	---	Currently not used
5	YMAX	inches	Maximum y-coordinate for all shot lines
6	YMIN	inches	Minimum y-coordinate for all shot lines
7	ZMAX	inches	Maximum z-coordinate for all shot lines
8	ZMIN	inches	Minimum z-coordinate for all shot lines
9	NOCOMP	---	Number of components in the target model

FIGURE 6-17. Program PEAKAY Binary Output, File 1, Viewing Plane.



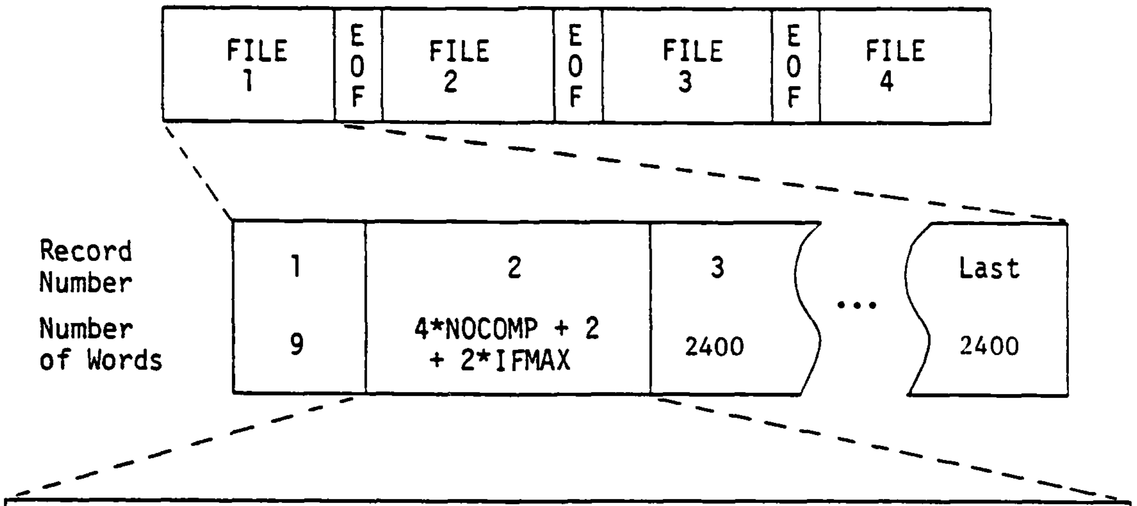
Record Number 2			
Word	Parameter	Units	Definition
1	ICOMP(1)	---	Component identification number for first component
2	IFG(1)	---	Criticality flag for first component =0: noncritical ≠0: critical
3	IY(1)	---	System number for first component
4	IU(1)	---	Vulnerability flag for first component =0: singly vulnerable =1: multiply vulnerable
	. . .		
	ICOMP(NOCOMP)	---	Component identification number for NOCOMPth component
	IFG(NOCOMP)	---	Criticality flag for the NOCOMPth component

FIGURE 6-18. Program PEAKAY Binary Output, File 1, Components and Flux (Page 1 of 3).



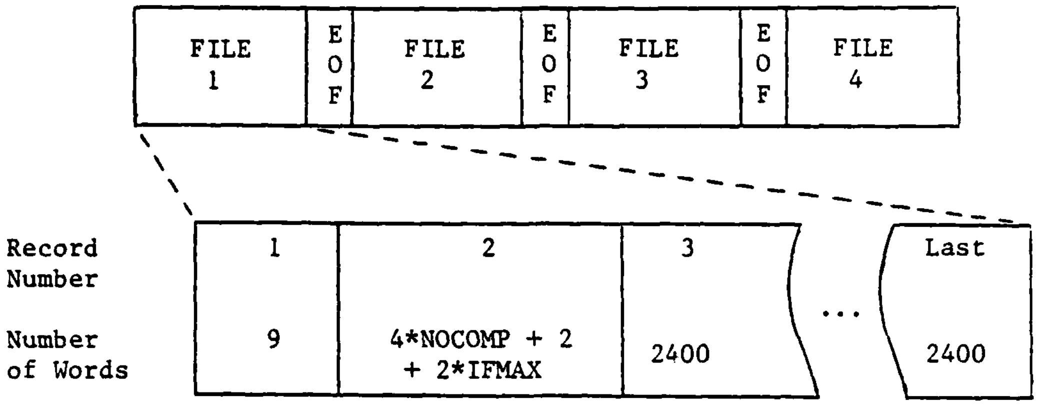
Record Number 2			
Word	Parameter	Units	Definition
	IY(NOCOMP)	---	System number for NOCOMP*th* component
4*NOCOMP	IU(NOCOMP)	---	Vulnerability flag for NOCOMP*th* component
	IRVRS	---	Shot line reverse flag =0: normal shot lines =1: reversed shot lines
4*NOCOMP+2	IFMAX	---	Number of points in the flux distribution table
	FTIM(1)	seconds	First time argument for the flux distribution
	FXCM(1)	watts/cm^2	Flux at time FTIM(1)
	FTIM(2)	seconds	Second time argument for the flux distribution

FIGURE 6-18. Program PEAKAY Binary Output, File 1, Components and Flux (Page 2 of 3).



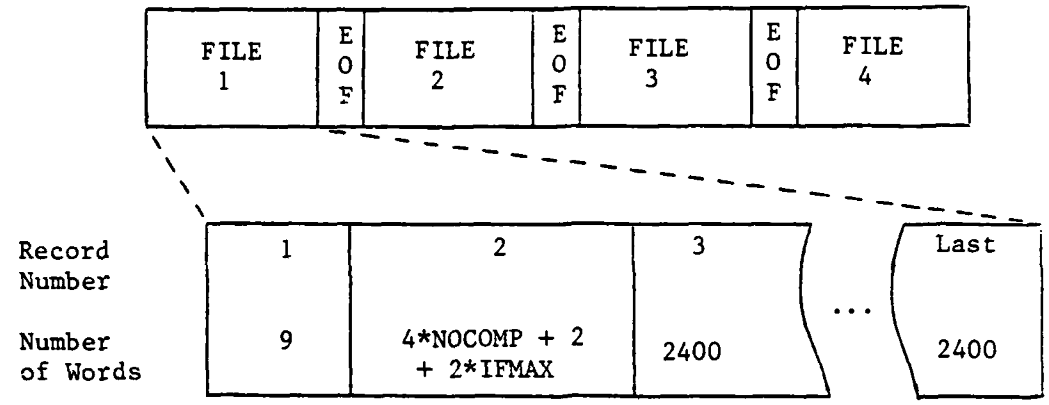
Record Number 2			
Word	Parameter	Units	Definition
	FXCM(2)	watts/cm^2	Flux at time FTIM(2)
	. . .		
	FTIM(IFMAX)	seconds	Last time argument for the flux distribution
4*NOCOMP +2+2*IFMAX	FXCM(IFMAX)	watts/cm^2	Flux at time FTIM(IFMAX)

FIGURE 6-18. Program PEAKAY Binary Output, File 1, Components and Flux (Page 3 of 3).



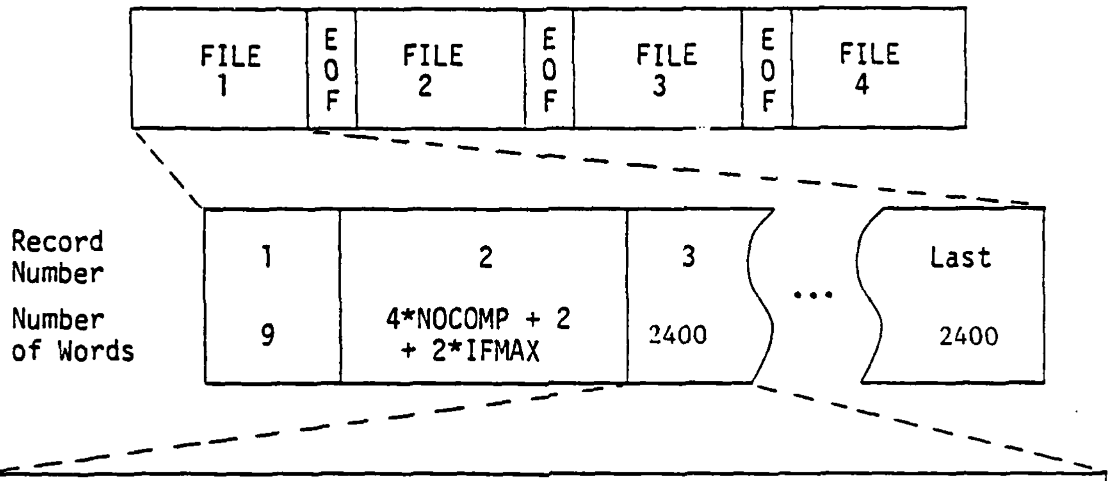
Record Number 3 - Last			
Word	Parameter	Units	Definition
1	R1(1,1)	inches seconds	y-coordinate for each new shot line, otherwise the time needed to penetrate to depth DEPTH(1)
2	R1(2,1)	inches seconds	z-coordinate for each new shot line, otherwise the time needed to penetrate to depth DEPTH(2)
3	R1(3,1)	seconds	the time needed to penetrate to depth DEPTH(3)
	. . .		
10	R1(10,1)	seconds	the time needed to penetrate to depth DEPTH(10)
11	R1(1,2)	inches seconds	y-coordinate for each new shot line, otherwise the time needed to penetrate to depth DEPTH(1)
12	R1(2,2)	inches seconds	z-coordinate for each new shot line, otherwise the time needed to penetrate to depth DEPTH(2)
	. . .		

FIGURE 6-19. Program PEAKAY Binary Output, File 1, Encounter Times (Page 1 of 3).



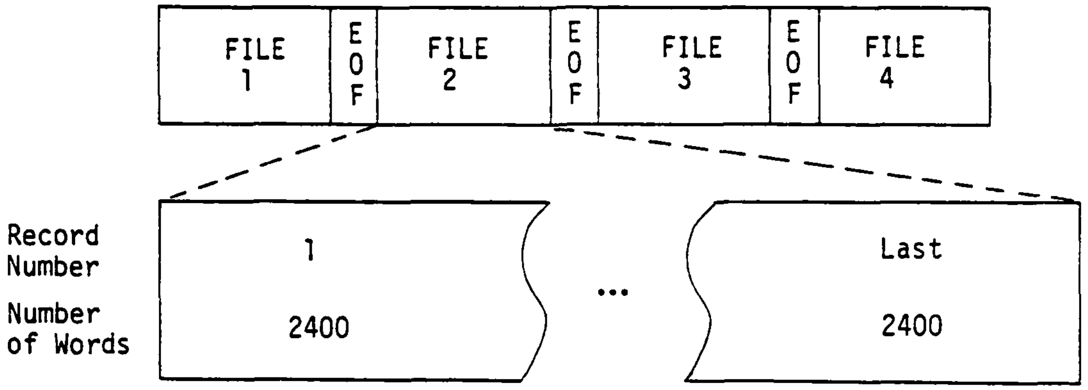
Record Number 3 - Last			
Word	Parameter	Units	Definition
2000	R1(10,200)	seconds	the time needed to penetrate to depth DEPTH(10)
2001	R3(1)	--- seconds	Equals 0.00 for each new shot line, otherwise the time needed to perforate the encountered component
2002	R3(2)	--- seconds	Equals 0.00 for each new shot line, otherwise the time needed to perforate the encountered component
	. . .		
2200	R3(200)	--- seconds	Equals 0.00 for each new shot line, otherwise the time needed to perforate the encountered component
2201	I1(1)	---	Number of encounters for each new shot line, otherwise the encountered component number, equals 9999.0 to indicate end of view

FIGURE 6-19. Program PEAKAY Binary Output, File 1, Encounter Times (Page 2 of 3).



Record Number 3 - Last			
Word	**Parameter**	**Units**	**Definition**
2201	I1(2)	---	Number of encounters for each new shot line, otherwise the encountered component number, equals 9999.0 to indicate end-of-view
	⋮		
2400	I1(200)	---	Number of encounters for each new shot line, otherwise the encountered component number, equals 9999.0 to indicate end-of-view

FIGURE 6-19. Program PEAKAY Binary Output, File 1, Encounter Times (Page 3 of 3).



Record Number 1 - Last			
Word	Parameter	Units	Definition
1	SHW(1,1)	inches	y-coordinate on the viewing plane for the first shot line
2	SHW(2,1)	inches	z-coordinate on the viewing plane for the first shot line
3	SHW(3,1)	---	Pk at first time increment for first shot line, = -9.0 if no critical encounters on shot line = -9999.0 if end-of-view
.	.		
.	.		
.	.		
12	SHW(12,1)	---	Pk at tenth time for first shot line = -9.0 if no critical encounters on shot line = -9999.0 if end-of-view
13	SHW(1,2)	inches	y-coordinate on the viewing plane for the second shot line
14	SHW(2,2)	inches	z-coordinate on the viewing plane for the second shot line

FIGURE 6-20. Program PEAKAY Binary Output, File 2, Shot Line Pk's (Page 1 of 2).

```
 NUMBER OF COMPS.=   10      IRVRS=    0     FACTOR FOR CONVERSION TO INCHES= .1000E+01

NO. OF POINTS IN FLUX DISTRIBUTION =  . 5

BEGIN TIME      END TIME       FLUX
       .00          2.00      2000.00
      2.00          4.00      4000.00
      4.00          6.00      6000.00
      6.00          8.00      8000.00
      9.00         10.00     10000.00
     10.00    **********     10000.00

NO TIMAX VALUES WERE READ IN ... FLUX TIME POINTS WILL BE USED
```

FIGURE 6-28. Program PRERD Output, Flux and No Time Intervals.

```
THE FOLLOWING DATA HAS BEEN READ AND CHECKED FOR ERRORS.
 AN ASTERISK IN THE LEFTMOST COLUMN INDICATES A POSSIBLE ERROR IN THAT LINE.
 THE CHARACTER < IS USED TO POINT TO THE POSSIBLY ERRONEOUS ITEM.

ICOMP     IEG        MAT       ITAB       IANA      TINIT     THINEL       RHOF        IY         IU       NOPNTS     CARD
       DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)  PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)
       DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)  PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)

    1      0         1         0         1       20.00       .00       .99       1         0          2         1
          .00       .00       .00      1.00
    2      1         0         5         2       20.00       .00       .99       1         0          2         2
          .10       .00       .10      1.00
    3      1         1         0         1       20.00       .00       .99       1         0          2         3
          .00       .00       .06      1.00
  999      1         0         4         1       20.00       .00       .99       1         0          2         4
          .00       .00       .00      1.00
 1001      2        11         0         2       20.00       .00       .10       2         0          2         5
          .10       .00       .10      1.00
 1002      2         0         4         1       20.00       .00       .90       2         0          2         6
          .05       .00       .50      1.00
 1101      1        11         0         2       20.00       .00       .10       2         0          2         7
          .10       .00       .10      1.00
 1102      1         0         4         1       20.00       .00       .90       2         0          2         8
          .05       .00       .50      1.00
 2001      3        10         0         2       20.00       .00       .99       3         0          2         9
          .00       .00       .10      1.00
 2101      3        11         0         1       20.00       .00       .10       3         0          2        10
          .10       .00       .15      1.00
 2102      1        10         0         1       20.00       .00       .99       3         0          2        11
          .01       .00       .10      1.00
 3001      1         0         5         2      400.00       .00       .06       4         0          2        12
          .00       .00       .10      1.00
 4001      1         0         5         2       20.00       .00       .99       5         0          2        13
          .02       .00       .22      1.00
```

FIGURE 6-29. Program PRERD Output, Errors Flagged.

```
CRITICAL COMPONENT SUMMARY

ICOMP    IFG       MAT       ITAB      IANA      TINIT     THINFL    RHOF      IY        IU         NOPNTS     CARD
         DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)  PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)
         DEPTH(6)  PKVAL(6)  DEPTH(7,  PKVAL(7)  DEPTH(8)  PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)

    2    1         0         5         2          20.00       .00       .99    1         0          2          2
          .10       .00       .10      1.00
    3    1         1         0         1          20.00       .00       .99    1         0          2          3
          .00       .00       .06      1.00
  999    1         0         4         1          20.00       .00       .99    1         0          2          4
          .00       .00       .00      1.00
 1101    1        11         0         2          20.00       .00       .10    2         0          2          7
          .10       .00       .10      1.00
 1102    1         0         4         1          20.00       .00       .90    2         0          2          8
          .05       .00       .50      1.00
 2102    1        10         0         1          20.00       .00       .99    3         0          2         11
          .01       .00       .10      1.00
 3001    1         0         5         2         400.00       .00       .06    4         0          2         12
          .00       .00       .10      1.00
 4001    1         0         5         2          20.00       .00       .99    5         0          2         13
          .02       .00       .22      1.00

THERE ARE    8 CRITICAL COMPONENTS HAVING CRITICALITY FLAG    1

 1001    2        11         0         2          20.00       .00       .10    2         0          2          5
          .10       .00       .10      1.00
 1002    2         0         4         1          20.00       .00       .90    2         0          2          6
          .05       .00       .50      1.00

THERE ARE    2 CRITICAL COMPONENTS HAVING CRITICALITY FLAG    2

 2001    3        10         0         2          20.00       .00       .99    3         0          2          9
          .00       .00       .10      1.00
 2101    3        11         0         1          20.00       .00       .10    3         0          2         10
          .10       .00       .15      1.00

THERE ARE    2 CRITICAL COMPONENTS HAVING CRITICALITY FLAG    3
```

FIGURE 6-30. Program PRERD Output, Critical Component Summary.

```
COMPONENT LISTING BY MATERIAL TYPE

ICOMP     IFG        MAT        ITAB       IANA       TINIT      THINFL     RHOF       IY         IU         NOPNTS     CARD
        DEPTH(1)   PKVAL(1)   DEPTH(2)   PKVAL(2)   DEPTH(3)   PKVAL(3)   DEPTH(4)   PKVAL(4)   DEPTH(5)   PKVAL(5)
        DEPTH(6)   PKVAL(6)   DEPTH(7)   PKVAL(7)   DEPTH(8)   PKVAL(8)   DEPTH(9)   PKVAL(9)   DEPTH(10)  PKVAL(10)

   1      0          1          0          1        20.00        .00        .99        1          0          2          1
           .00        .00        .00       1.00
   3      1          1          0          1        20.00        .00        .99        1          0          2          3
           .00        .00        .06       1.00

THERE ARE    2 COMPONENTS OF MATERIAL TYPE   1

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   2

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   3

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   4

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   5

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   6

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   7

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   8

THERE ARE    0 COMPONENTS OF MATERIAL TYPE   9

2001      3         10          0          2        20.00        .00        .99        3          0          2          9
           .00        .00        .10       1.00
2102      1         10          0          1        20.00        .00        .99        3          0          2         11
           .01        .00        .10       1.00

THERE ARE    2 COMPONENTS OF MATERIAL TYPE  10

1001      2         11          0          2        20.00        .00        .10        2          0          2          5
           .10        .00        .10       1.00
1101      1         11          0          2        20.00        .00        .10        2          0          2          7
           .10        .00        .10       1.00
2101      3         11          0          1        20.00        .00        .10        3          0          2         10
           .10        .00        .15       1.00
```

FIGURE 6-31. Program PRERD Output, Component Listing by Material Type.

```
COMPONENT LISTING BY TABLE-LOOKUP INDEX

ICOMP    IFG       MAT       ITAB      IANA      TINIT     THINFL    RHOF      IY        IU         NOPNTS     CARD
       DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)  PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)
       DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)  PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   1

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   2

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   3

   999     1         0         4         1        20.00       .00       .99       1         0          2          4
           .00       .00       .00      1.00
  1002     2         0         4         1        20.00       .00       .90       2         0          2          6
           .05       .00       .50      1.00
  1102     1         0         4         1        20.00       .00       .90       2         0          2          8
           .05       .00       .50      1.00

THERE ARE    3 COMPONENTS HAVING TABLE INDEX   4

     2     1         0         5         2        20.00       .00       .99       1         0          2          2
           .10       .00       .10      1.00
  3001     1         0         5         2       400.00       .00       .06       4         0          2         12
           .00       .00       .10      1.00
  4001     1         0         5         2        20.00       .00       .99       5         0          2         13
           .02       .00       .22      1.00

THERE ARE    3 COMPONENTS HAVING TABLE INDEX   5

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   6

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   7

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   8

THERE ARE    0 COMPONENTS HAVING TABLE INDEX   9
```

FIGURE 6-32. Program PRERD Output, Component Listing by Table Look-up Index.

```
COMPONENT LISTING BY ANALYSIS TYPE

ICOMP     IFG        MAT        ITAH       TANA       TINIT      THINEL     RHOF       IY         IU         NOPNTS       CARD
        DEPTH(1)   PKVAL(1)   DEPTH(2)   PKVAL(2)   DEPTH(3)   PKVAL(3)   DEPTH(4)   PKVAL(4)   DEPTH(5)   PKVAL(5)
        DEPTH(6)   PKVAL(6)   DEPTH(7)   PKVAL(7)   DEPTH(8)   PKVAL(8)   DEPTH(9)   PKVAL(9)   DEPTH(10)  PKVAL(10)

THERE ARE     0 COMPONENTS OF ANALYSIS TYPE    0

    1     0          1          0          1          20.00       .00        .99        1          0          2            1
          .00        .00        .00        1.00
    3     1          1          0          1          20.00       .00        .99        1          0          2            3
          .00        .00        .06        1.00
  999     1          0          4          1          20.00       .00        .99        1          0          2            4
          .00        .00        .00        1.00
 1002     2          0          4          1          20.00       .00        .90        2          0          2            6
          .05        .00        .50        1.00
 1102     1          0          4          1          20.00       .00        .90        2          0          2            8
          .05        .00        .50        1.00
 2101     3          11         0          1          20.00       .00        .10        3          0          2           10
          .10        .00        .15        1.00
 2102     1          10         0          1          20.00       .00        .99        3          0          2           11
          .01        .00        .10        1.00

THERE ARE     7 COMPONENTS OF ANALYSIS TYPE    1

    2     1          0          5          2          20.00       .00        .99        1          0          2            2
          .10        .00        .10        1.00
 1001     2          11         0          2          20.00       .00        .10        2          0          2            5
          .10        .00        .10        1.00
 1101     1          11         0          2          20.00       .00        .10        2          0          2            7
          .10        .00        .10        1.00
 2001     3          10         0          2          20.00       .00        .99        3          0          2            9
          .00        .00        .10        1.00
 3001     1          0          5          2         400.00       .00        .06        4          0          2           12
          .00        .00        .10        1.00
 4001     1          0          5          2          20.00       .00        .99        5          0          2           13
          .02        .00        .22        1.00

THERE ARE     6 COMPONENTS OF ANALYSIS TYPE    2
```

FIGURE 6-33. Program PRERD Output, Component Listing by Analysis Type.

Change 1

```
COMPONENT LISTING BY SYSTEM NUMBER

ICOMP     IFG        MAT       ITAR       IANA       TINIT      THINFL     RHOF        IY         IU        NUPNTS     CARD
        DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)  PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)
        DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)  PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)

THERE ARE    0 COMPONENTS IN SYSTEM NUMBER    0

    1     0          1         0          1        20.00       .00       .99         1          0          2          1
         .00        .00       .00       1.00
    2     1          0         5          2        20.00       .00       .99         1          0          2          2
         .10        .00       .10       1.00
    3     1          1         0          1        20.00       .00       .99         1          0          2          3
         .00        .00       .06       1.00
  999     1          0         4          1        20.00       .00       .99         1          0          2          4
         .00        .00       .00       1.00

THERE ARE    4 COMPONENTS IN SYSTEM NUMBER    1

 1001     2         11         0          2        20.00       .00       .10         2          0          2          5
         .10        .00       .10       1.00
 1002     2          0         4          1        20.00       .00       .90         2          0          2          6
         .05        .00       .50       1.00
 1101     1         11         0          2        20.00       .00       .10         2          0          2          7
         .10        .00       .10       1.00
 1102     1          0         4          1        20.00       .00       .90         2          0          2          8
         .05        .00       .50       1.00

THERE ARE    4 COMPONENTS IN SYSTEM NUMBER    2

 2001     3         10         0          2        20.00       .00       .99         3          0          2          9
         .00        .00       .10       1.00
 2101     3         11         0          1        20.00       .00       .10         3          0          2         10
         .10        .00       .15       1.00
 2102     1         10         0          1        20.00       .00       .99         3          0          2         11
         .01        .00       .10       1.00

THERE ARE    3 COMPONENTS IN SYSTEM NUMBER    3

 3001     1          0         5          2       400.00       .00       .06         4          0          2         12
         .00        .00       .10       1.00

THERE ARE    1 COMPONENTS IN SYSTEM NUMBER    4

 4001     1          0         5          2        20.00       .00       .99         5          0          2         13
         .02        .00       .22       1.00

THERE ARE    1 COMPONENTS IN SYSTEM NUMBER    5
```

FIGURE 6-34. Program PRERD Output, Component Listing by System Number.

```
NUMBER OF COMPS.=   10      INVRS=    0     FACTOR FOR CONVERSION TO INCHES= .1000E+01

    ***** INPUT ERROR ***** IFMAX =  .       27

NO. OF POINTS IN FLUX DISTRIBUTION =   27

BEGIN TIME     END TIME       FLUX
       .00         2.00     2000.00
      2.00         4.00     4000.00
      4.00         6.00     6000.00
      6.00         8.00     8000.00
      8.00        10.00    10000.00
     10.00        11.00    11000.00
     11.00        12.00    12000.00
     12.00        13.00    13000.00
     13.00        14.00    15000.00
     14.00        15.00    20000.00
     15.00        16.00    21000.00
     16.00        17.00    25000.00
     17.00        18.00    30000.00
     18.00        19.00    35000.00
     19.00        20.00    40000.00
     20.00        21.00    45000.00
     21.00        22.00    50000.00
     22.00        23.00    55000.00
     23.00        24.00    60000.00
     24.00        25.00    60500.00
     25.00        26.00    61000.00
     26.00        27.00    10000.00
     27.00        28.00     5000.00
     28.00        29.00     1000.00
     29.00        30.00         .00
     30.00   **********         .00

    ***** INPUT ERROR ***** FLUX = ,    .61E+05

    ***** INPUT ERROR ***** FLUX = ,    .61E+05

    ***** INPUT ERROR ***** FLUX = ,    .00E+00

    ***** INPUT ERROR ***** NTMAX =          26

ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING
```

FIGURE 6-35. Program PRERD Output, Error Messages.

```
ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING
```

FIGURE 6-36. Program PRERD Output, Error Message (NOCOMP>498).

PROGRAM QKPK OUTPUT

Execution of Program QKPK produces two output files. Formatted output is printed on Logical Unit Number 6. Figures 6-37 through 6-41 are examples of the formatted output that is generated during a normal execution of Program QKPK. Figures 6-42 through 6-47 are examples of error messages which may be printed during program execution. A binary file containing penetration times for each encounter is written on Logical Unit Number 2. Figures 6-48 through 6-50 display the formats for each record on the binary file.

QKPK Printed Output

The first page of QKPK printed output, Figure 6-37, lists the number of components, shot line reverse flag, input lengths conversion factor, and the flux distribution table. The times in the flux table are in seconds and the flux levels are in watts per square centimeter. This page is printed during execution of Subroutine RDATA.

Figure 6-38 is an example of the second page of QKPK formatted output. It is printed during execution of Program QKPK after the component information arrays have been sorted into ascending order. The first ten columns correspond to the parameters on cards QK6, QK6A, and QK6B from the QKPK data deck. The column labeled "CRD" gives the component subscript number in the component identification number array. This number is used to identify components in the QKPK binary output file.

The type of output shown in Figure 6-39 is printed only if the value of the flag IECHO is set to a value other than zero. These data are a complete copy of the binary output file and may fill several pages of printout. Figures 6-48 through 6-50 contain definitions for those parameters appearing in the binary output file.

Figure 6-40 is an example of the next page of QKPK output which shows the number of critical shot lines that reach their maximum Pk during each time interval. The time intervals are those specified on cards QK8 in the QKPK data deck. The times in the table are expressed in seconds and the flux levels are expressed in watts per square centimeter. The last time interval represents the value for all times greater than the last start time. Only critical shot lines are included in the breakdown. A critical shot line must encounter at least one critical component.

An example of the last page of QKPK output is shown in Figure 6-41. WRITE statements in the Main program unit are used to print this page. It lists any components in an encounter whose LOS thickness is less than either DEPTH(1) or DEPTH(NOPNTS). If the thickness is less than DEPTH(1), the encounter Pk is 0.00. If thickness is less than DEPTH(NOPNTS), the Pk is less than 1.00. However, a thickness equal to DEPTH(NOPNTS) may or may not equal 1.00; this depends on the value assigned to PKVAL(NOPNTS). Entries in this list indicate possible errors in the selection of DEPTH values.

Figure 6-42 presents the fatal error message printed during execution of Subroutine RDATA whenever the number of components in the target model (QKPK) data deck) exceeds array dimensions. The current program has a limit of 498 components.

The fatal error message shown as Figure 6-43 is also printed during Subroutine RDATA execution. This error occurs whenever a flux level less than or equal to 0.0 or greater than 60000.0 watts per square centimeter is entered from the data card deck.

Figure 6-44 contains a warning message printed whenever an encountered component number from the LOS file has no match in the component information arrays. This warning message is printed during execution of Subroutine BSRCH. When components are included in the target model for the shot line generating program but not in the QKPK data deck, default component characteristics are assigned as follows:

1. Components with identification numbers less than 1000 default to a 2024 AL noncritical component with no warning message printed.

2. Components with identification numbers between 1000 and 6999 or greater than 8000 default to a 7075 AL noncritical component with a warning message printed as shown in Figure 6-44.

3. Components with identification numbers between 7000 and 7999 default to a 7075 AL noncritical component with no warning message printed.

Figure 6-45 shows a non-fatal error message printed whenever Subroutine TABLE is invoked for a component whose table look-up code is less than or equal to 0 or greater than 9. A penetration rate equal to 0.00 is assigned and control of execution returns to Subroutine RATE. The warning message includes the component number and the erroneous table look-up code.

Figure 6-46 shows a fatal error message which is printed during Subroutine TABLE execution. This error occurs when a component has a table look-up code equal to 7 and a weapon flux level less than 374.0 watts per square centimeter.

If the Main QKPK program unit detects a shot line with more than 100 encounters, an error message (Figure 6-47) is printed and program execution stops. This prevents bounds violations in the encounter data arrays.

QKPK Binary Output

Parameters and their definitions for each record of the binary penetration times file are listed on Figures 6-48 through 6-50. The first record contains nine parameter values and describes the viewing plane. The second record contains values for the component information arrays, the shot line reverse flag, and the flux distribution table. The third and all subsequent records contain the encounter penetration times. The last record on this file contains an end-of-view flag, parameter I1 equal to 9999. This file is used as the binary input file for Program PEAKAY.

```
 NUMBER OF COMPS.=   10       IRVRS=    0      FACTOR FOR CONVERSION TO INCHES= .1000E+01

NO. OF POINTS IN FLUX DISTRIBUTION =    5

BEGIN TIME      END TIME        FLUX
       .00          2.00     2000.00
      2.00          4.00     4000.00
      4.00          6.00     6000.00
      6.00          8.00     8000.00
      8.00         10.00    10000.00
     10.00    **********    10000.00
```

FIGURE 6-37. Program QKPK Output, Flux Distribution.

```
VALUES USED FOR THIS RUN

ICOMP    IFG       MAT       ITAB      IANA      TINIT     THINFL    RHOF      IY        IU         CARD
       DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)  PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)
       DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)  PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)

    1    0         1         0         1        20.00      .00       .99       1         0          1
         .00       .00       .00      1.00
    2    1         0         5         2        20.00      .00       .99       1         0          2
         .10       .00       .10      1.00
    3    1         1         0         1        20.00      .00       .99       1         0          3
         .00       .00       .06      1.00
  999    1         0         4         1        20.00      .00       .99       1         0          4
         .00       .00       .00      1.00
 1001    2         11        0         2        20.00      .00       .10       2         0          5
         .10       .00       .10      1.00
 1002    2         0         4         1        20.00      .00       .90       2         0          6
         .05       .00       .23       .25        .31      .25       .33      .50       .37        .50
         .41       .75       .43       .80        .45      .90       .48      .95       .50       1.00

 1101    1         11        0         2        20.00      .00       .10       2         0          7
         .10       .00       .10      1.00
 1102    1         0         4         1        20.00      .00       .90       2         0          8
         .05       .00       .50      1.00
 2001    3         10        0         2        20.00      .00       .99       3         0          9
         .00       .00       .10      1.00
 2101    3         11        0         1        20.00      .00       .10       3         0          10
         .10       .00       .15      1.00
 2102    1         10        0         1        20.00      .00       .99       3         0          11
         .00       .00       .10      1.00
 3001    1         0         5         2       400.00      .00       .06       4         0          12
         .00       .00       .10      1.00
 4001    1         0         5         2        20.00      .00       .99       5         0          13
         .02       .00       .06       .20        .10      .40       .14      .60       .18        .80
         .22      1.00
```

FIGURE 6-38. Program QKPK output, Component Information.

```
THE FOLLOWING IS AN ECHO OF THE DATA WRITTEN ON THE QKPK OUTPUT FILE ( LOGICAL UNIT  2 )

AZ, FL, GRID, IDVEH, YMAX, YMIN, ZMAX, ZMIN, NOCOMP
 45.00000  30.00000    .25000       1   6.52250   -.75222   4.03358   -.79111      13

ICOMP(I), IFG(I), IY(I), IU(I), NOPNTS(I), (PKVAL(J,I)J=1,NUPNTS(I)),I=1,NOCOMP
     1      0      1      0      2     .00      1.00
     2      1      1      0      2     .00      1.00
     3      1      1      0      2     .00      1.00
   999      1      1      0      2     .00      1.00
  1001      2      2      0      2     .00      1.00
  1002      2      2      0     10     .00       .25      .25      .50      .50      .75      .80      .90      .95     1.00
  1101      1      2      0      2     .00      1.00
  1102      1      2      0      2     .00      1.00
  2001      3      3      0      2     .00      1.00
  2101      3      3      0      2     .00      1.00
  2102      1      3      0      2     .00      1.00
  3001      1      4      0      2     .00      1.00
  4001      1      5      0      6     .00       .20      .40      .60      .80     1.00

IRVRS, IFMAX, (FTIM(I), FXCM(I), I=1,IFMAX          0       5
     2.000  2000.000           4.000  4000.000           6.000  6000.000           8.000  8000.000
    10.000 10000.000

PKTIME(J,I), PLS(I), II(I), J=1,10, I=1,200
    .12500   -.62500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     1
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
    .37500   -.62500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     1
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
    .62500   -.62500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     0
    .87500   -.62500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     0
   -.12500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     1
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
    .12500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     1
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
    .37500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     2
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
   5.62645   5.62645    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000   2.44892     2
    .62500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     3
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
   4.59597   4.59597    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000   3.64634     2
   4.02610   4.78562   5.54514   6.30465   7.06417   7.82369    .00000    .00000    .00000    .00000   4.82861    13
    .87500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     2
   5.33251   5.33251    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000   4.73563     2
   5.17183   6.04421   6.91660   7.78899   8.66138   9.53376    .00000    .00000    .00000    .00000   6.14937    13
   1.12500   -.37500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     2
   6.13257   6.13257    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000   5.65953     2
   6.67666   8.71093  10.74520  12.77946  14.81373  16.84799    .00000    .00000    .00000    .00000   9.14937    13
   -.37500   -.12500    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000     1
    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .00000    .10383     4
```

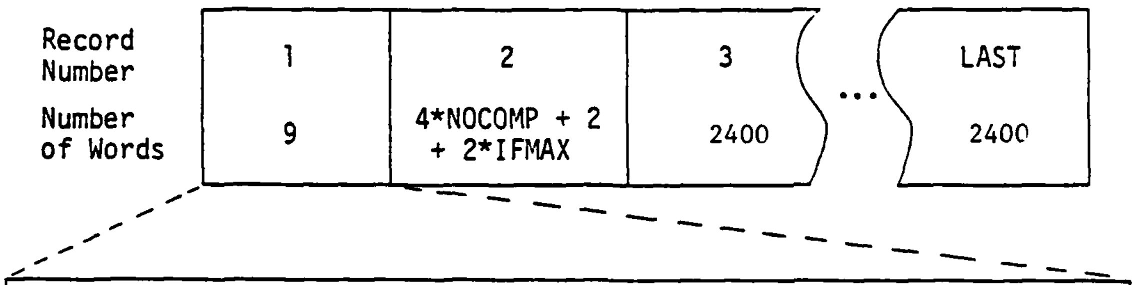
FIGURE 6-39. Program QKPK Output, Optional Echo of Binary File Output.

```
TOTAL NUMBER OF NON-CRITICAL SHOTLINES =    102

BREAKDOWN OF CRITICAL SHOTLINES BY TIME NEEDED TO REACH MAXIMUM PK

BEGIN TIME      END TIME      BEGIN FLUX      END FLUX      NO. OF SHOTLINES
       .00           .10        2000.00        2000.00                     0
       .10           .20        2000.00        2000.00                     0
       .20           .40        2000.00        2000.00                    30
       .40           .60        2000.00        2000.00                     0
       .60           .80        2000.00        2000.00                     3
       .80          1.00        2000.00        2000.00                     1
      1.00          1.90        2000.00        2000.00                    36
      1.90          3.90        2000.00        4000.00                    26
      3.90          4.90        4000.00        6000.00                    16
      4.90          5.90        6000.00        6000.00                     0
      5.90    **********        6000.00       10000.00                     0
                          TOTAL NUMBER OF CRITICAL SHOTLINES =           114

TOTAL NUMBER OF SHOTLINES =       216
```

FIGURE 6-40. Program QKPK Output, Shot Line Breakdown.

```
NENC =       101 WHICH IS GREATER THAN 100
...PROGRAM HALTING
```

FIGURE 6-47. Program QKPK Output, More Than 100 Encounters on a Shot Line.



Record Number 1			
Word	Parameter	Units	Definition
1	AZ	degrees	Attack azimuth angle
2	EL	degrees	Attack elevation angle
3	GRID	inches	Grid cell size
4	IDVEH	---	Currently not used
5	YMAX	inches	Maximum y-coordinate for all shot lines
6	YMIN	inches	Minimum y-coordinate for all shot lines
7	ZMAX	inches	Maximum z-coordinate for all shot lines
8	ZMIN	inches	Minimum z-coordinate for all shot lines
9	NOCOMP	---	Number of components in the target model

FIGURE 6-48. Program QKPK Binary Output, Viewing Plane Description.

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 2			
Word	Parameter	Units	Definition
1	ICOMP(1)	---	Component identification number for first component
2	IFG(1)	---	Criticality flag for the first component =0: noncritical ≠0: critical
3	IY(1)	---	System number for the first component
4	IU(1)	---	Vulnerability flag for first component =0: singly vulnerable =1: multiply vulnerable
5	ICOMP(2)	---	Component identification number for second component
. . .	. . .		
.	ICOMP(NOCOMP)	---	Component identification number for NOCOMP*th* component
.	IFG(NOCOMP)	---	Criticality flag for the NOCOMP*th* component
.	IY(NOCOMP)	---	System number for the NOCOMP*th* component

FIGURE 6-49. Program QKPK Binary Output, Component Data and Flux Distribution (Page 1 of 2).

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 2			
Word	Parameter	Units	Definition
4*NOCOMP	IU(NOCOMP)	---	Vulnerability flag for NOCOMPth component
	IRVRS	---	Shot line reverse flag =0: normal encounter order =1: reversed encounter order
4*NOCOMP+2	IFMAX	---	Number of points in the flux distribution table
.	FTIM(1)	seconds	First time argument in the flux distribution
.	FXCM(1)	watts/cm^2	Weapon intensity, flux at time FTIM(1)
.	FTIM(2)	seconds	Second time argument in the flux distribution
	FXCM(2)	watts/cm^2	Weapon intensity, flux at time FTIM(2)
	. . .		
	FTIM(IFMAX)	seconds	Last time argument in the flux distribution
4*NOCOMP +2+2*IFMAX	FXCM(IFMAX)	watts/cm^2	Weapon intensity, flux at time FTIM(IFMAX)

FIGURE 6-49. Program QKPK Binary Output, Component Data and Flux Distribution (Page 2 of 2).

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 - Last

Word	Parameter	Units	Definition
1	PKTIME(1,1)	inches	y-coordinate on the viewing plane for a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(1) in an encounter with component I1(1)
2	PKTIME(2,1)	inches	z-coordinate on the viewing plane for a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(2) in an encounter with component I1(1)
3	PKTIME(3,1)	seconds	Time needed to pentrate from the start of the shot line to depth DEPTH(3) in an encounter with component I1(1)
.	.		.
.	.		.
.	.		.
10	PKTIME(10,1)	seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(10) in an encounter with component I1(1)

FIGURE 6-50. Program QKPK Binary Output, Penetration Times (Page 1 of 4)

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 - Last			
Word	Parameter	Units	Definition
11	PKTIME(1,2)	inches	y-coordinate on the viewing plane for a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(1) in an encounter with component I1(2)
12	PKTIME(2,2)	inches	z-coordinate on the viewing plane for a new shot line
		or	
		seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(2) in an encounter with component I1(2)
. . .	. . .		. . .
20	PKTIME(10,2)	seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(10) in an encounter with component I1(2)
. . .	. . .		. . .
2000	PKTIME(10,200)	seconds	Time needed to penetrate from the start of the shot line to depth DEPTH(10) in an encounter with component I1(200)

FIGURE 6-50. Program QKPK Binary Output, Penetration Times (Page 2 of 4)

Record Number	1	2	3	...	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 - Last			
Word	Parameter	Units	Definition
401	PLS(1)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(1)
402	PLS(2)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(2)
⋮	⋮		⋮
600	PLS(200)	---	Equals 0.00 for a new shot line
		or	or
		seconds	Time needed to penetrate from the start of the shot line through the LOS thickness of component I1(200)

FIGURE 6-50. Program QKPK Binary Output, Penetration Times (Page 3 of 4).

Record Number	1	2	3	. . .	LAST
Number of Words	9	4*NOCOMP + 2 + 2*IFMAX	2400		2400

Record Number 3 - Last			
Word	**Parameter**	**Units**	**Definition**
2201	I1(1)	---	Number of encounters on a new shot line
		or	or
		---	Component in the encounter with penetration times PKTIME(1,1), PKTIME(2,1), . . ., PKTIME(10,1), and PLS(1) (the subscript in the ICOMP array for the corresponding component identification number
		or	or
		---	Equals 9999 to indicate the end of view
. . .	. . .		. . .
2400	I1(200)	---	Number of encounters on a new shot line
		or	or
		---	Component in the encounter with penetration times PKTIME(1,200), PKTIME(2,200), . . ., PKTIME(10,200), and PLS(200) (the subscript in the ICOMP array for the corresponding component identification number
		or	or
		---	Equals 9999 to indicate the end of view

FIGURE 6-50. Program QKPK Binary Output, Penetration Times (Page 4 of 4)

```
      SUBROUTINE BSRCH(ICOMP,NOCOMP,ITC)
C
C          SUBROUTINE TO DO A BINARY SEARCH ON INTEGER ARRAY ICOMP(NOCOMP)
C               WHICH WAS SORTED INTO ASCENDING ORDER, AND RETURN THE
C               LOCATION J WHERE THE MATCH ON ICOMP IS FOUND
C
      COMMON   /PROP/ ALP,RHO,CP,TMLT,XLAMBD,RATE,JCOMP,J,DP,XK,TVAP,
     $              CPL,ITER,T
      COMMON   /LUNITS/ IRD,IWR,IIN,IOUT
      DIMENSION ICOMP(ITC)
C
      IBEG = 0
      IEND = NOCOMP + 1
   10 CONTINUE
        J = (IBEG + IEND) / 2
        IF (J .NE. IBEG) THEN
          IF (ICOMP(J) .LT. JCOMP) THEN
            IBEG = J
          ELSE IF (ICOMP(J) .GT. JCOMP) THEN
            IEND = J
          END IF
        ELSE
          IF (JCOMP .LT. 1000) THEN
            J = ITC - 1
          ELSE IF ((JCOMP .GE. 7000) .AND. (JCOMP .LE. 7999)) THEN
            J = ITC
          ELSE
            WRITE (IWR,100) JCOMP
            J = ITC
          END IF
          RETURN
        END IF
      IF (ICOMP(J) .NE. JCOMP) GO TO 10
C
      RETURN
C
  100 FORMAT (1X,I10,' WAS NOT FOUND IN LIST...DEFAULT TO 7075 AL')
      END
```

Intentionally Left Blank

```
CF $ STEP F
CF   REWIND TAPE AND REWRITE FIRST RECORD WITH MIN AND MAX VIEWING PLANE
CF * COORDINATES
C
      REWIND IOUT
      WRITE (IOUT) AZM,ELEV,GRID,LL,YMX,YMN,ZMX,ZMN,R
      IF (IECHO.NE.0) WRITE (IWR,1003) AZM,ELEV,GRID,LL,YMX,YMN,ZMX,ZMN,
     1 R
      REWIND IOUT
      WRITE (IWR,240) YMX,YMN,ZMX,ZMN
      WRITE (IWR,245)
C
CF   EXIT STOP
C
      STOP
C
  180 FORMAT (F7.1,7X,F9.3,F9.3,I3,18X,F6.1,2X,F6.1)
  190 FORMAT (3(I4,F7.2,F5.1,I3,F7.2))
  200 FORMAT (2(I4,F7.2,7X,F6.1,I3,F7.2))
  210 FORMAT (1X,F6.1,F7.1,3X,F8.2,8X,I3)
  220 FORMAT (I5,10A6)
  225 FORMAT (15H1NO. OF VIEWS =,I5/1H0,10A6)
  230 FORMAT (2(5X,E15.8),30X,E10.3)
  235 FORMAT (10H0AZIMUTH =,F10.2,13H  ELEVATION =,F10.2,
     1  13H  GRID SIZE =,F10.2)
  240 FORMAT (7H0YMAX =,F10.2,8H  YMIN =,F10.2/7H0ZMAX =,F10.2,
     2  8H  ZMIN =,F10.2)
  245 FORMAT (1H1)
 1000 FORMAT (1H1,69HTHE FOLLOWING IS AN ECHO OF THE DATA WRITTEN ON THE
     1 MAGIC OUTPUT FILE//1H0,30HA, EL, GRID, LL, X, X, X, X, R/
     2  1H ,3F10.3,I10,5F10.2)
 1001 FORMAT (1H0,67HAX(I), AY(I), AZ(I), ISRI(I), IECO(I), ITH(I), IIB(
     1,IOB(I), I=1,170/(1H ,3F10.3,5I10))
 1002 FORMAT (1H0,34HXX, XX, XX, II, XX, XX, XX, XX, XX/1H ,3F10.3,I10,
     1 5F10.3)
 1003 FORMAT (1H0,42HAZM, ELEV, GRID, LL, YMX, YMN, ZMX, ZMN, R/
     1  1H ,3F10.3,I10,5F10.3/1H ,5X,73H(THE RECORD ABOVE ACTUALLY OVER-
     2WRITES THE VERY FIRST RECORD ON THE FILE)/1H0,32HEND OF ECHO OF MA
     1GIC OUTPUT FILE/1H1)
C
CF   FINISH
C
      END
      SUBROUTINE FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,
     $                 DEPTH,PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
C
C           SUBROUTINE TO SORT ARRAY ICOMP (OF LENGTH NOCOMP) INTO
C                      ASCENDING ORDER AND MAINTAIN CORRESPONDENCE
C                      OF OTHER ARRAYS
C
      INTEGER IANA  (ITC),
     $        ICOMP (ITC),
     $        IFG   (ITC),
     $        ITAB  (ITC),
     $        IU    (ITC),
     $        IY    (ITC),
     $        MAT   (ITC),
     $        NOPNTS(ITC),
     $        PKNBR
      REAL    BUFFR1(50),
     $        BUFFR2(50),
     $        DEPTH (PKNBR,ITC),
     $        PKVAL (PKNBR,ITC),
     $        RHOF  (ITC),
     $        THINFL(ITC),
     $        TINIT (ITC)
C
      M = NOCOMP
   10 CONTINUE
        M = M / 2
C
        IF (M .LT. 1) GO TO 50
        K = NOCOMP - M
C
        DO 40 J = 1,K
          I = J
   20     CONTINUE
            IM = I + M
            IF (ICOMP(I) .LE. ICOMP(IM)) GO TO 40
C
C       SWITCH
C
            NO = ICOMP(I)
            N1 = MAT(I)
            N2 = IFG(I)
            N3 = ITAB(I)
            N4 = IANA(I)
            N5 = IY(I)
            N6 = IU(I)
            N7 = NOPNTS(I)
            S1 = TINIT(I)
            DO 30 MM = 1,PKNBR
              BUFFR1(MM) = DEPTH(MM,I)
              BUFFR2(MM) = PKVAL(MM,I)
```

```
30        CONTINUE
          S4 = THINFL(I)
          S5 = RHOF(I)
          ICOMP(I) = ICOMP(IM)
          MAT(I) = MAT(IM)
          IFG(I) = IFG(IM)
          ITAB(I) = ITAB(IM)
          IANA(I) = IANA(IM)
          IY(I) = IY(IM)
          IU(I) = IU(IM)
          NOPNTS(I) = NOPNTS(IM)
          TINIT(I) = TINIT(IM)
          DO 34 MM = 1,PKNBR
            DEPTH(MM,I) = DEPTH(MM,IM)
            PKVAL(MM,I) = PKVAL(MM,IM)
34        CONTINUE
          THINFL(I) = THINFL(IM)
          RHOF(I) = RHOF(IM)
          ICOMP(IM) = N0
          MAT(IM) = N1
          IFG(IM) = N2
          ITAB(IM) = N3
          IANA(IM) = N4
          IY(IM) = N5
          IU(IM) = N6
          NOPNTS(IM) = N7
          TINIT(IM) = S1
          DO 38 MM = 1,PKNBR
            DEPTH(MM,IM) = BUFFP1(MM)
            PKVAL(MM,IM) = BUFFP2(MM)
38        CONTINUE
          THINFL(IM) = S4
          RHOF(IM) = S5
          I = I - M
        IF (I .GE. 1) GO TO 20
40    CONTINUE
   GO TO 10
C
50 CONTINUE
C
   RETURN
   END
```

A-9

Change 1

```
C             PROGRAM PEAKAY
C
CF START PEAKAY
CF TITLE                      QKLOOK:  PROGRAM PEAKAY
CF ENTER PEAKAY
C
      INTEGER    AVFLAG
      COMMON     R1(10,200),R3(200),I1(200),PKTIME(10,100),
     $           PKVAL(10,100),NOPNTS(500),PLS(100),
     $           NAME(100),IST,NSHT,NENC,ICOMP(500),SHW(12,200),
     $           NOCOMP,COMPAV(500,10),GRID,TIMES(10),NTIME,IY(500),
     $           PMULT(10,10),IU(500),PAREA(500)
      COMMON    /ONE/ IFG(500),PRNT(4,500),AVFLAG
      COMMON    /LUNITS/ IRD,IWR,IIN,IOUT
      DIMENSION PKNK(500),TVA(10),TVAM(10),ISET(10),SAREA(10),
     $           FTIM(25),FXCM(25)
      DATA       IRD,IWR,IIN,IOUT1,IOUT2,IOUT3,IOUT4 /7,8,2,10,11,12,13/
      DATA       PMULT /100*0./
      DATA       IST,NUM,NSHT /3*0/
      DATA       PAREA /500*0./
      DATA       (COMPAV(I,1),I=1,500) /500*0./
      DATA       (COMPAV(I,2),I=1,500) /500*0./
      DATA       (COMPAV(I,3),I=1,500) /500*0./
      DATA       (COMPAV(I,4),I=1,500) /500*0./
      DATA       (COMPAV(I,5),I=1,500) /500*0./
      DATA       (COMPAV(I,6),I=1,500) /500*0./
      DATA       (COMPAV(I,7),I=1,500) /500*0./
      DATA       (COMPAV(I,8),I=1,500) /500*0./
      DATA       (COMPAV(I,9),I=1,500) /500*0./
      DATA       (COMPAV(I,10),I=1,500) /500*0./
      DATA       (PRNT(1,I),I=1,500) /500*0./
      DATA       (PRNT(2,I),I=1,500) /500*0./
      DATA       (PRNT(3,I),I=1,500) /500*0./
      DATA       (PRNT(4,I),I=1,500) /500*0./
      DATA       ITOTL /0/
      DATA       TVA /10*0./
      DATA       SAREA /10*0./
      DATA       ITC /500/
      DATA       PCV /90./
      DATA       SQFSQM /0.09290304/
C
      OPEN (IRD,FILE='PKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
      OPEN (IWR,FILE='PKPRINT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
      OPEN (IIN,FILE='QKPKTAPEOUT',FORM='UNFORMATTED',RECFM='VARIABLE')
      OPEN (IOUT1,FILE='PKTAPEOUT1',FORM='UNFORMATTED',RECFM='VARIABLE')
      OPEN (IOUT2,FILE='PKTAPEOUT2',FORM='UNFORMATTED',RECFM='VARIABLE')
      OPEN (IOUT3,FILE='PKTAPEOUT3',FORM='UNFORMATTED',RECFM='VARIABLE')
      OPEN (IOUT4,FILE='PKTAPEOUT4',FORM='UNFORMATTED',RECFM='VARIABLE')
C
C             THIS PROGRAM PERFORMS THE COMPONENT PENETRATION
C             AND COMPUTES COMPONENT AND SYSTEM VULNERABLE
```

```
C              AREAS...AND SHOT LINE PK'S
C
C              ITC  IS THE DIMENSION OF THE COMPONENT ARRAYS
C              PCV  IS THE PER CENT VULNERABLE AREA USED FOR AN OUTPUT
C              SQFSQM  IS THE CONVERSION FROM SQ. FEET TO SQ.METERS
C
      REWIND ITN
C
CF $ STEP A
CF   READ-CARDS CONTAINING UP TO 10 TIME INTERVALS
C
      READ (IRD,1003) NTIME,AVFLAG
      READ (IRD,1001) (TIMES(J),J=1,NTIME)
C
CF   READ VIEWING PLANE DATA, COMPONENT DATA, AND FLUX TABLE
C
      READ (IIN) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
      READ (IIN) (ICOMP(I),IFG(I),IY(I),IU(I),NOPNTS(I),
     $  (PKVAL(J,I),J=1,NOPNTS(I)),I=1,NOCOMP),IRVRS,IFMAX,
     $  (FTIM(I),FXCM(I),I=1,IFMAX)
C
C              ITC MUST BE AT LEAST TWO GREATER THAN NOCOMP TO ALLOW
C              FOR THE DEFAULTS TO 2024 AL AND 7075 AL
C
CF   IF TOO MANY COMPONENTS? THEN 500

      IF (ITC .GE. NOCOMP+2) THEN
C
CF $ STEP B
CF   PRINT THE TIME INTERVALS AND THE FLUX DISTRIBUTION
C
        WRITE (IWR,4) NTIME,(TIMES(J),J=1,NTIME)
        RVRS = IRVRS
        WRITE (IWR,8) RVRS
        WRITE (IWR,800) IFMAX
        TE = 0.
C
        DO 820 I = 1,IFMAX
          TB = TE
          TE = FTIM(I)
          WRITE (IWR,810) TB,TE,FXCM(I)
  820   CONTINUE
C
C              LAST FLUX ALSO APPLIES FOR ALL TIMES BEYOND
C              THE LAST TIME
C
        TB = TE
        TE = 1.E30
        WRITE (IWR,810) TB,TE,FXCM(IFMAX)
C
CF   WRITE FILE-1   A COMPLETE COPY OF THE BINARY INPUT FILE
```

```
C
         WRITE (IOUT1) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
         WRITE (IOUT1) (ICOMP(I),IFG(I),IY(I),IU(I),I=1,NOCOMP),IRVRS,
      $    IFMAX,(FTIM(I),FXCM(I),I=1,IFMAX)
C
   10    CONTINUE
           READ (IIN) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
           WRITE (IOUT1) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
         IF (I1(200) .NE. 9999) GO TO 10
C
         END FILE IOUT1
         REWIND IIN
         GRID = GRID * GRID / 144.
C
C           GRID IS NOW THE AREA OF ONE GRID CELL IN SQUARE FEET
C
         READ (IIN)
         READ (IIN)
C
C           NTIME IS THE NUMBER OF DWELL TIMES TO CONSIDER -- TIMES
C           IS THE LIST OF THEM IN ASCENDING ORDER
C
         MAXTIM = NTIME + 2
C
CF $ STEP C
CF   READ THE NEXT RECORD OF THE TIME FILE
C
  100    CONTINUE
           READ (IIN) ((R1(I,J),I=1,10),R3(J),I1(J),J=1,200)
C
CF $ STEP D
CF   LOOP TO PROCESS EACH SET OF INPUT ARRAY ELEMENTS
C
  101      CONTINUE
           DO 110 J = 1,200
C
CF   IF END OF VIEW? THEN 5000
C
             IF (I1(J) .EQ. 9999) GO TO 5000
C
C           COMPONENT EQUAL TO 9999 SIGNALS END OF VIEW
C
             IF (IST .LE. 0) THEN
C
C           ITOTL COUNTS TOTAL SHOT LINES FOR VIEW
CF      STORE SHOTLINE ENCOUNTER DATA
C
               ITOTL = ITOTL + 1
               NENC = I1(J)
               NSHT = NSHT + 1
               SHW(1,NSHT) = R1(1,J)
```

```
                 SHW(2,NSHT) = R1(2,J)
                 IF (NENC .NE. 0) THEN
                   IST = NSHT
                   NUM = 0
                 ELSE
C
C          SET -9 TO SIGNAL NO CRITICAL COMPONENTS ALONG THIS SHOT LINE
C
C
                   DO 112 MM = 3,MAXTIM
                     SHW(MM,NSHT) = -9.
  112              CONTINUE
                 END IF
               ELSE
C
                 NUM = NUM + 1
                 DO 114 K = 1,10
                   PKTIME(K,NUM) = R1(K,J)
  114            CONTINUE
                 PLS(NUM) = R3(J)
                 NAME(NUM) = I1(J)
C
CF   IF LAST ENCOUNTER ON THIS SHOT LINE? THEN * ELSE 110
C
                 IF (NUM .GE. NENC) THEN
C
CF $ STEP E
CF            INCREMENT THE PRESENTED AREA FOR EACH CRITICAL
CF *          COMPONENT AND EACH SYSTEM ON THIS SHOT LINE
C
                   DO 115 I = 1,10
                     ISET(I) = 0
  115              CONTINUE
C
                   DO 118 I = 1,NENC
                     II = NAME(I)
                     LL = IY(II)
                     IF (LL .NE. 0) ISET(LL) = 1
                     IF (I .NE. NENC) THEN
                       DO 116 I3 = I+1,NENC
                         IF (II .EQ. NAME(I3)) GO TO 118
  116                  CONTINUE
                     END IF
                     PAREA(II) = PAREA(II) + 1.
  118              CONTINUE
C
                   DO 119 I = 1,10
                     SAREA(I) = SAREA(I) + ISET(I)
  119              CONTINUE
C
CF         EXECUTE PENT PERFORMS SHOT LINE PENETRATION
CF *     COMPUTING PK'S AT EACH TIME INCREMENT
C
                   CALL PFNT
                   IST = 0
C
C             ACCUMULATE PK'S FOR TOTAL TARGET VULNERABLE AREA
C
                   DO 107 II = 1,NTIME
                     TVA(II) = TVA(II) + SHW(II+2,NSHT)
  107              CONTINUE
                 END IF
               END IF
C
CF $ STEP F
CF    IF SHOT LINE PK ARRAYS ARE FULL? THEN * ELSE 110
C
               IF (NSHT .EQ. 200) THEN
C**** WRITE (IWR,1002) ((SHW(II,MM),II=1,3),MM=1,200)
C
CF    WRITE FILE-2  THE SHOT LINE PK ARRAYS
C
                 WRITE (IOUT2) ((SHW(II,MM),II=1,12),MM=1,200)
                 NSHT=0
               END IF
C
CF $ STEP G
CF    IF LAST SET OF INPUT FROM TIME FILE RECORD? THEN 100 ELSE 101
C
  110      CONTINUE
C
         GO TO 100
C
C             SET -9999 TO SIGNAL END OF VIEW FOR PLOTIT
C
CF $ STEP H
CF    WRITE FILE-2  LAST RECORD OF SHOT LINE ARRAYS WITH END OF VIEW FLAG
C
 5000    CONTINUE
         NSHT = NSHT + 1
C
         DO 5010 K = 3,MAXTIM
           SHW(K,NSHT) = -9999.
 5010    CONTINUE
C
         WRITE (IOUT2) ((SHW(II,MM),II=1,12),MM=1,200)
         END FILE IOUT2
C
C**** WRITE (IWR,1002) ((SHW(II,MM),II=1,3),MM=1,200)
C
CF    COMPUTE VULNERABLE AREAS FROM COMPONENT PK'S
CF    PRINT PRESENTED AND VULNERABLE AREA SUMMARIES
```

```
C
          WRITE (IWR,7001) PCV
          JQQ = 0
          FRV = PCV * 0.01
C
          DO 8000 MQ = 1,NOCOMP
            IF (IFG(MQ) .NE. 0) THEN
              IF(PRNT(3,MQ) .LE. 0.) THEN
C
C               SAVE THE NAMES OF ALL COMPONENTS WHICH WERE NOT KILLED
C
                JQQ = JQQ + 1
                PKNK(JQQ) = ICOMP(MQ)
              END IF
C
C               DETERMINE TIME INTERVAL DURING WHICH COMPONENT'S
C               VULNERABLE AREA REACHED PCV% OF ITS PRESENTED AREA
C
              IF (PAREA(MQ) .GT. 0.) THEN
C
                DO 6010 K = 1,NTIME
                  IF (COMPAV(MQ,K)/PAREA(MQ) .GE. FRV) THEN
                    IF (K .GT. 1) THEN
                      TB = TIMES(K-1)
                    ELSE
                      TB = 0
                    END IF
                    WRITE (IWR,7002) ICOMP(MQ),(PRNT(JJ,MQ),JJ=1,4),TB,
     $                TIMES(K)
                    GO TO 8000
                  END IF
 6010           CONTINUE
C
              END IF
C
              WRITE (IWR,7002) ICOMP(MQ),(PRNT(JJ,MQ),JJ=1,4)
            END IF
 8000     CONTINUE
C
C             CONVERT COMPONENT PK'S TO VULNERABLE AREAS
C
          DO 305 J = 1,NOCOMP
            PAREA(J) = PAREA(J) * GRID
            DO 300 K = 1,NTIME
              COMPAV(J,K) = COMPAV(J,K) * GRID
  300       CONTINUE
  305     CONTINUE
C
C        CONVERT TOTAL TARGET PK'S AND SYSTEM PK'S TO VULNERABLE AREAS
C
          DO 315 K = 1,NTIME
          TVA(K) = TVA(K) * GRID
          TVAM(K) = TVA(K) * SQFSQM
          DO 310 J = 1,10
            PMULT(J,K) = PMULT(J,K) * GRID
  310     CONTINUE
  315   CONTINUE
C
C
CF    WRITE FILE-3  FLUX TABLE, TIME INTERVALS, AND VULNERABLE AREAS
C
        IF (AVFLAG .EQ. 1) THEN
          WRITE (IWR,1004) (TIMES(K),K=1,NTIME)
        ELSE
          WRITE (IWR,1007) (TIMES(K),K=1,NTIME)
        END IF
        WRITE (IOUT3) AZ,EL,IFMAX,(FTIM(I),FXCM(I),I=1,IFMAX),RVRS,
     $    NOCOMP,NTIME,(TIMES(I),I=1,NTIME)
C
        DO 320 J = 1,NOCOMP
          IF (IFG(J) .NE. 0) THEN
            WRITE (IWR,1005) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
            WRITE (IOUT3) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
          END IF
  320   CONTINUE
C
        END FILE IOUT3
C
        TOTL = ITOTL*GRID
        WRITE (IWR,7005) TOTL,ITOTL
        WRITE (IWR,7006) (TVA(K),K=1,NTIME)
        WRITE (IWR,1008) (TIMES(K),K=1,NTIME)
C
        DO 325 K = 1,10
          SAREA(K) = SAREA(K) * GRID
          WRITE (IOUT4) SAREA(K),K,(PMULT(K,J),J=1,NTIME)
          WRITE (IWR,1005) K,SAREA(K),(PMULT(K,J),J=1,NTIME)
  325   CONTINUE
C
C
CF    WRITE FILE-4 SYSTEM AND TOTAL TARGET VULNERABLE AREAS AT EACH TIME
C
        WRITE (IOUT4) JQQ,(PKNK(NCN),NCN=1,JQQ)
C
C           CONVERT SQUARE FEET TO SQUARE METERS
C
        DO 360 J = 1,NOCOMP
          PAREA(J) = PAREA(J) * SQFSQM
          DO 350 K = 1,NTIME
            COMPAV(J,K) = COMPAV(J,K) * SQFSQM
  350     CONTINUE
  360   CONTINUE
```

```
C
          IF (AVFLAG .EQ. 1) THEN
            WRITE (IWR,1006) (TIMES(K),K=1,NTIME)
          ELSE
            WRITE (IWR,1011) (TIMES(K),K=1,NTIME)
          END IF
C
          DO 370 J = 1,NOCOMP
            IF (IFG(J) .NE. 0)
     $        WRITE (IWR,1005) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
  370     CONTINUE
C
          TOTL = TOTL * SQFSQM
          WRITE (IWR,7007) TOTL,ITOTL
          WRITE (IWR,7008) (TVAM(K),K=1,NTIME)
          WRITE (IWR,1009) (TIMES(K),K=1,NTIME)
C
          DO 400 K = 1,10
            SAREA(K) = SAREA(K) * SQFSQM
C
            DO 380 J = 1,NTIME
              PMULT(K,J) = PMULT(K,J) * SQFSQM
  380       CONTINUE
C
            WRITE (IOUT4) SAREA(K),K,(PMULT(K,J),J=1,NTIME)
            WRITE (IWR,1005) K,SAREA(K),(PMULT(K,J),J=1,NTIME)
  400     CONTINUE
C
          WRITE (IWR,7011)
C
          DO 330 J = 1,NOCOMP
            IF ((IFG(J) .NE. 0) .AND. (PAREA(J) .LE. 0.))
     $        WRITE (IWR,7012) ICOMP(J)
  330     CONTINUE
C
          WRITE (IWR,7013)
C
          DO 340 J = 1,NOCOMP
            IF ((IFG(J) .NE. 0) .AND. (PRNT(3,J) .LE. 0.)) THEN
              ASQF = PAREA(J)/SQFSQM
              WRITE (IWR,1010) ICOMP(J),ASQF,PAREA(J)
            END IF
  340     CONTINUE
C
          WRITE (IOUT4) (TVA(I),TVAM(I),I=1,NTIME)
        ELSE
C
CE    PRINT FATAL ERROR MESSAGE, TOO MANY COMPONENTS
C
          WRITE (IWR,7014)
        END IF
```

```
C
CF    EXIT STOP
C
       STOP
C
    4 FORMAT (' NTIME=',I5,' STEPS'/10X,' TIME STEPS ARE:',10F10.3)
    8 FORMAT (10X,' IRVRS=',F5.0)
  800 FORMAT (' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5/
     $   ' BEGIN TIME',5X,'END TIME',7X,'FLUX')
  810 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
 1001 FORMAT (10F8.2)
 1002 FORMAT (5(3F6.2,2X))
 1003 FORMAT (2I5)
 1004 FORMAT ('1 PRESENTED AREA AND TRUE COMPONENT VULERABLF AREAS ',
     $  '(SQ. FEET) PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
 1005 FORMAT (I5,F10.5,5X,10F10.5)
 1006 FORMAT ('1 PRESENTED AREA AND TRUE COMPONENT VULERABLE AREAS ',
     $  '(SQ. METERS) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
 1007 FORMAT ('1 PRESENTED AREA AND INCREMENTAL COMPONENT VULERABLE ',
     $  'AREAS (SQ. FEET) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
 1008 FORMAT ('1 PRESENTED AREA AND SYSTEM VULERABLE AREAS (SQ. FEET) ',
     $  'PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
 1009 FORMAT ('1 PRESENTED AREA AND SYSTEM VULERABLE AREAS (SQ. METERS)',
     $  ' PER TIME INCREMENT'//' TIME INCREMENTS',4X,10F10.2/)
 1010 FORMAT (1X,I6,9X,2E12.5)
 1011 FORMAT ('1 PRESENTED AREA AND INCREMENTAL COMPONENT VULERABLE ',
     $  'AREAS (SQ. METERS) PER TIME INCREMENT'//' TIME INCREMENTS',4X,
     $  10F10.2/)
 7001 FORMAT('1',4X,'COMP. NO    Y-COORD.    Z-COORD.     MAX. PK      ',
     $  ' TIME',5X,'TIME INTERVAL DURING WHICH VULERABLE AREA REACHES',
     $  F4.0,'%'/66X,'OF THE PRESENTED AREA'/)
 7002 FORMAT (I11,2X,2E12.4,F11.4,F12.4,10X,F12.4,' TO',F12.4,' SECONDS')
 7005 FORMAT (' TOTAL TARGET PRESENTED AREA (SQ. FEET) =',F10.5,
     $  5X,'(',I6,' TOTAL SHOTLINES)')
 7006 FORMAT (' TOTAL TARGET VULNERABLE AREA (SQ. FEET) PER ',
     $  'TIME INCREMENT'//(20X,10F10.5))
 7007 FORMAT (' TOTAL TARGET PRESENTED AREA (SQ. METERS) =',F10.5,
     $  5X,'(',I6,' TOTAL SHOTLINES)')
 7008 FORMAT (' TOTAL TARGET VULNERABLE AREA (SQ. METERS) PER ',
     $  'TIME INCREMENT'//(20X,10F10.5))
 7011 FORMAT ('1 CRITICAL COMPONENTS WHOSE PRESENTED AREA EQUALS ZERO',
     $' ARE:')
 7012 FORMAT(1X,I10)
 7013 FORMAT ('1 CRITICAL COMPONENTS WHOSE VULNERABLE AREA EQUALS ZERO',
     $' ARE:'/' COMPONENT      PRESENTED AREA (SQ. FEET AND SQ. METERS)')
 7014 FORMAT (' ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING')
C
CF    FINISH
C
       END
```

Change 1

```
      SUBROUTINE PENT
C
C             PKC ALLOWS 100 INTERSECTIONS PER SHOT LINE
C             SHW IS SET UP TO SAVE Y,Z,1ST TIME,...,10TH TIME PK
C             PER 200 SHOT LINES
C
CF START PENT
CF TITLE                    QKLOOK:  SUBROUTINE PENT
CF ENTER PENT
C
      INTEGER   AVFLAG
      DIMENSION PKC(100),PKS(10)
      COMMON    R1(10,200),R3(200),T1(200),PKTIME(10,100),
     $          PKVAL(10,100),NOPNTS(500),PLS(100),
     $          LOC(100),IST,NSHT,NENC,ICOMP(500),SHW(12,200),
     $          NOCOMP,COMPAV(500,10),GRID,TIMES(10),NTIME,IY(500),
     $          PMULT(10,10),IU(500),PAREA(500)
      COMMON   /ONE/ IFG(500),PRNT(4,500),AVFLAG
      LOGICAL   SNGSYS,MULSYS,TXSET
C
C            THIS SUBROUTINE IS ENTERED WITH THE NECESSARY INFO FOR ONE
C            SHOT LINE,  ENERGY IS PASSED ALONG THIS SHOT LINE FOR ALL
C            TIME DURATIONS DESIRED, STARTING WITH THE SHORTEST.  A TIME
C            HISTORY IS KEPT SO THAT FOR EACH SUCCESSIVE TIME IT IS NEC-
C            ESSARY ONLY TO PENETRATE THOSE COMPONENTS NOT PREVIOUSLY
C            BREACHED BY SHORTER TIMES.
C
C            PKTIME(1)  IS THE TIME NEEDED TO REACH PK-MIN FOR THIS
C                       COMPONENT, STARTING FROM THE BEGINNING OF THE SHOT
C                       LINE.
C            PKTIME(I)  IS THE TIME NEEDED TO REACH AN INTERMEDIATE PK
C                       FOR THIS COMPONENT, STARTING FROM THE BEGINNING
C                       OF THE SHOT LINE. (I IS IN THE INTERVAL [1,N-1]).
C            PKTIME(N)  IS THE TIME NEEDED TO REACH PK-MAX FOR THIS
C                       COMPONENT, STARTING FROM THE BEGINNING OF THE SHOT
C                       LINE. (N IS IN THE INTERVAL [2,10]).
C            PLS        IS THE TIME NEEDED TO PERFORATE THIS COMPONENT,
C                       STARTING FROM THE BEGINNING OF THE SHOT LINE.
C            TBEG       IS THE LOWER BOUND OF THE TIME INTERVAL (I.E., IT
C                       REPRESENTS THE TIME ALREADY USED UP)
C            TEND       IS THE DURATION TIME FROM ARRAY TIMES WHICH IS
C                       CURRENTLY BEING PROCESSED.
C            PK1        IS THE TIME AT WHICH THE MAXIMUM POSSIBLE PK FOR
C                       THIS COMPONENT WILL BE ACHIEVED.  (NOTE  PK=1.0
C                       MAY NOT BE POSSIBLE SINCE PLS MAY BE LESS THAN
C                       PKTIME(N) OR PKVAL(N) MAY BE LESS THAN 1.0).
C            PK2        IS THE MINIMUM TIME NEEDED TO GET A NON-ZERO PK
C                       FOR THIS COMPONENT IN THE INTERVAL TBEG TO TEND.
C            PK3        IS THE TIME NEEDED TO GET MAXIMUM PK FOR THIS
C                       COMPONENT IN THE INTERVAL TBEG TO TEND.
      PMX = 0.
      TBEG = 0.
      NEND = 1
C
      DO 10 II=1,10
        PKS(II) = 0.
   10 CONTINUE
C
CF $ STEP A
CF   TIME PENETRATION LOOP,  ITERATE FOR EVERY TIME INTERVAL
C
   11 CONTINUE
      DO 6000 NTIM=1,NTIME
        TEND = TIMES(NTIM)
        NBEG = NEND
C
CF $ STEP B
CF   ENCOUNTER LOOP,  ITERATE FOR EACH REMAINING ENCOUNTER
C
   12   CONTINUE
        DO 1000 II=NBEG,NENC
          NEND = II
          PK = 0.
          PKC(II) = 0.
          SPK = 0.
          I = LOC(II)
          L = IY(I)
          SNGSYS = (L .NE. 0) .AND. (IU(I) .EQ. 0)
          MULSYS = (L .NE. 0) .AND. (IU(I) .EQ. 1)
C
CF   IF INTERVAL ENDS BEFORE MIN PENETRATION? THEN 1900
C
          IF (TEND .LT. PKTIME(1,II)) GO TO 1900
          IF (PLS(II) .GE. PKTIME(1,II)) THEN
            IF (PKVAL(NOPNTS(I),I) .EQ. 1.0) THEN
              PK1 = AMIN1(PKTIME(NOPNTS(I),II),PLS(II))
            ELSE
              PK1 = PLS(II)
            END IF
C
CF   IF MAX PENETRATION ENDED IN LAST INTERVAL? THEN 1000
C
            IF (TBEG .LT. PK1) THEN
              IX = 1
              IXSET = .FALSE.
   20         CONTINUE
                IF ((IX .LT. NOPNTS(I)) .AND.
     $            (PKTIME(IX,II) .LT. TEND)) THEN
                  IX = IX + 1
                ELSE
                  IXSET = .TRUE.
                END IF
```

```
                  IF (.NOT. IXSET) GO TO 20
                  PK2 = PKTIME(IX-1,II)
                  PK3 = AMIN1(TEND,PK1)
C
CF    COMPUTE PK FOR THIS ENCOUNTER
C
                  IF ((PK3 .LE. PKTIME(NOPNTS(I),II)) .AND.
     $               (PK2 .NE. PKTIME(IX,II))) THEN
                     PK = (PK3 - PK2) / (PKTIME(IX,II) - PK2) *
     $                 (PKVAL(IX,I) - PKVAL(IX-1,I)) + PKVAL(IX-1,I)
                  ELSE
                     PK = PKVAL(IX,I)
                  END IF
C
C              PK IS NOW THE COMPONENT PK FOR THIS ENCOUNTER ONLY
C
C              SPK IS NOW THE SYSTEM PK FOR THIS ENCOUNTER ONLY
C
                  SPK = PK
                  IF (AVFLAG .EQ. 0) PK = PK * (1.0 - PMX)
                  PKC(II) = PK
                  IF (II .NE. 1) THEN
C
CF $ STEP C
CF    COMPUTE COMPONENT PK FOR ALL PREVIOUS ENCOUNTERS
C
C
C            IF THIS COMPONENT WAS HIT BEFORE ON THIS SHOT LINE,ADJUST
C           CURRENT HIT PK ASSUMING EVENT INDEPENDENCE
C
C             (PK FOR THIS ENCOUNTER MUST BE CONDITIONED BY PROB OF
C              SURVIVAL FOR ALL PREVIOUS ENCOUNTERS.)
C
                     DO 172 LM = II-1,1,-1
C
C              FIND LATEST ENCOUNTER OF THIS COMPONENT PRIOR TO
C              THE CURRENT ENCOUNTER
C
C              PKC(N) IS THE TOTAL PK FOR THIS COMPONENT (ON THIS SHOTLINE)
C              UP TO (AND INCLUDING) ENCOUNTER NO. N
C
                       IF (LOC(LM) .EQ. I) THEN
                         PK = PK * (1.0 - PKC(LM))
                         PKC(II) = PKC(LM) + PK
                         GO TO 175
                       END IF
  172                CONTINUE
                  END IF
C
  175             CONTINUE
C
```

```
C             PKS(N) IS THE TOTAL PK FOR SYSTEM NO. N (INCLUDING
C             SINGLY-VULNERABLE COMPONENTS ONLY).
C
C             SPK MUST BE ADJUSTED TO REFLECT ANY PREVIOUS ENCOUNTER
C             OF THIS SYSTEM ON THIS SHOT LINE
C
CF      ACCUMULATE SYSTEM PK FOR SINGLY VULNERABLE SYSTEMS ONLY
C
                IF (SNGSYS) SPK = SPK * (1.0 - PKS(L))
                IF (PKC(II) .GT. PRNT(3,I)) THEN
C
C            STORE MAX COMPONENT PK, LOCATION, AND TIME IN PRNT
C
                  PRNT(1,I) = SHW(1,IST)
                  PRNT(2,I) = SHW(2,IST)
                  PRNT(3,I) = PKC(II)
                  PRNT(4,I) = PK3
                ELSE IF ((PKC(II) .EQ. PRNT(3,I)) .AND.
     $            (PRNT(4,I) .GT. PK3)) THEN
                  PRNT(1,I) = SHW(1,IST)
                  PRNT(2,I) = SHW(2,IST)
                  PRNT(4,I) = PK3
                END IF
C
C            COMPAV HOLDS CUMULATIVE PK FOR EACH COMPONENT -- LATER
C                   WILL BE MULTIPLIED BY THE GRID AREA TO GIVE VULNERABLE
C                   AREA
C
CF $ STEP D
CF      IF ENCOUNTER ENDS IN LATER INTERVAL? THEN 1900
C
                IF (TEND .LT. PK1) GO TO 1900
C
CF            COMPONENT HAS BEEN COMPLETELY PROCESSED,
CF *          RECORD PK'S AND GO TO NEXT COMPONENT
C
                IF (SNGSYS) PKS(L) = PKS(L) + SPK
                TREG = PK3
C
                DO  500 NTI = NTIM,NTIMF
C
C             PMULT ACCUMULATES PK'S FOR SINGLY VULNERABLE MEMBERS
C             OF SYSTEMS ONLY
C
                  IF (SNGSYS) PMULT(L,NTI) = PMULT(L,NTI) + SPK
                  COMPAV(I,NTI) = COMPAV(I,NTI) + PK
 500            CONTINUE
C
C            IU(I) IS A MULTIPLY-VULNERABLE COMPONENT OF SYSTEM IY(I)
C            NOTE -- MULT.VUL. COMP. ARE NOT ADDED TO TOTAL SHOTLINE COUNT
C
```

```
                    IF ((.NOT. MULSYS) .AND. (AVFLAG .EQ. 1)) THEN
                      PMX = PMX + PK * (1.0 - PMX)
                    ELSE IF ((.NOT. MULSYS) .AND. (AVFLAG .EQ. 0)) THEN
                      PMX = PMX + PK
                    END IF
                  END IF
                END IF
C
CF $ STEP E
CF   IF ANOTHER ENCOUNTER ON THE SHOT LINE? THEN 12
C
 1000   CONTINUE
C
CF          ALL COMPONENTS HAVE BEEN PENETRATED
C           PMX STAYS THE SAME FOR ALL REMAINING DURATIONS.
C
        DO 1500 NTI = NTIM,NTIME
          SHW(NTI+2,IST) = PMX
 1500   CONTINUE
C
CF   EXIT RETURN
C
        RETURN
C
CF $ STEP F
CF          TIME DURATION IS COMPLETELY USED UP,
CF *        RECORD PK'S AND GO TO NEXT TIME INTERVAL
C
 1900   CONTINUE
        IF (.NOT. MULSYS) THEN
          IF (AVFLAG .EQ. 1) THEN
            SHW(NTIM+2,IST) = PMX + PK * (1.0 - PMX)
          ELSE
            SHW(NTIM+2,IST) = PMX + PK
          END IF
        ELSE
          SHW(NTIM+2,IST) = PMX
        END IF
        IF (SNGSYS) PMULT(L,NTIM) = PMULT(L,NTIM) + SPK
        COMPAV(I,NTIM) = COMPAV(I,NTIM) + PK
C
CF   IF ANOTHER TIME INTERVAL? THEN 11
C
 6000 CONTINUE
C
      RETURN
C
CF   EXIT RETURN
CF   FINISH
C
      END
```

```
C             PROGRAM PRERD
C
CF START PRERD
CF TITLE                        QKLOOK:  PROGRAM PRERD
CF ENTER PRERD
C
C       THIS PROGRAM READS THE QKLOOK INPUT DECK, CHECKS FOR POSSIBLE
C       ERRORS, AND PRINTS COMPONENT SUMMARIES.
C
        INTEGER     PKNBR
        COMMON      ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
       $            TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
       $            RHOF(500),IU(500),IY(500)
        COMMON      THINFL(500),NOCOMP
        COMMON     /LUNITS/ IRD,IWR
        COMMON     /SIZES/ ITC,IFX
        DIMENSION IE(34)
        DATA        IBLNK,ICHK,IASTER /' ',1H<,'*'/
        DATA        IRD,IWR /5,6/
        DATA        ITC,IFX,PKNBR /500,25,10/
C
        OPEN (IRD,FILE='QKPKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
        OPEN (IWR,FILE='PRERDOUT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
C
C             ITC  IS THE DIMENSION OF THE COMPONENT ARRAYS
C             IFX  IS THE DIMENSION OF THE FLUX ARRAYS
C
CF $ STEP A
CF   EXECUTE READIN  READS THE INPUT DECK AND TESTS FOR ARRAY BOUNDS
CF *                 VIOLATIONS
C
        CALL READIN
C
CF   EXECUTE FSORT  SORTS THE COMPONENT INFORMATION ARRAYS INTO
CF *                ASCENDING ORDER BY COMPONENT NUMBER
C
        CALL FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,DEPTH,
       $  PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
C
        WRITE (IWR,100)
        WRITE (IWR,101)
C
CF $ STEP B
CF   TEST FOR DUPLICATE COMPONENT ID NUMBERS
C
        DO 110 J = 2,NOCOMP
          IF (ICOMP(J) .EQ. ICOMP(J-1)) THEN
            ICOMP(J) = -IABS(ICOMP(J))
            ICOMP(J-1) = -IABS(ICOMP(J-1))
          END IF
  110 CONTINUE
```

Change 1

```
C
CF $ STEP C
CF    ITERATE FOR EVERY COMPONENT
C
       DO 199 J = 1,NOCOMP
         DO 120 I = 1,34
           IE(I) = IBLNK
  120    CONTINUE
C
CF    TEST FOR COMPONENT NUMBERS LESS THAN 1 OR GREATER THAN 9999
C
         IF (ICOMP(J) .LE. 0) IE(1) = ICHK
         ICOMP(J) = IABS(ICOMP(J))
         IF (ICOMP(J) .GT. 9999) IE(1) = ICHK
C
CF    TEST FOR CRITICALITY FLAG NOT EQUAL TO 0  1  2  OR 3
C
         IF ((IFG(J) .LT. 0) .OR. (IFG(J) .GT. 3)) IE(2) = ICHK
C
CF    TEST TABLE LOOK-UP CODE AND MATERIAL CODE
C
         IF ((ITAB(J) .EQ. 0) .AND. ((MAT(J) .LT. 1) .OR.
     $     (MAT(J) .GT. 11))) IE(3) = ICHK
         IF ((MAT(J) .EQ. 0) .AND. ((ITAB(J) .LT. 1) .OR.
     $     (ITAB(J) .GT. 9))) IE(4) = ICHK
         IF ((ITAB(J) .NE. 0) .AND. (MAT(J) .NE. 0)) THEN
           IE(3) = ICHK
           IE(4) = ICHK
         END IF
C
CF    TEST FOR ANALYSIS CODE NOT EQUAL TO 0  1  OR 2
C
         IF ((IANA(J) .LT. 0) .OR. (IANA(J) .GT. 2)) IE(5) = ICHK
C
CF    TEST FOR NEGATIVE OR ZERO INITIAL OPERATING TEMPERATURE
C
         IF (TINIT(J) .LE. 0.) IE(6) = ICHK
C
CF    TEST INFLUENCE MODE THICKNESS AND DENSITY FACTOR
C
         IF (THINFL(J) .LT. 0.) IE(9) = ICHK
         IF ((RHOF(J) .LT. 0.) .OR. (RHOF(J). GT. 1.)) IE(10) = ICHK
         IF (((THINFL(J) .GT. 0.) .AND. (RHOF(J) .GT. 0.)) .OR.
     $     ((THINFL(J) .LE. 0.) .AND. (RHOF(J) .LE. 0.))) THEN
           IE(9) = ICHK
           IE(10) = ICHK
         END IF
C
CF    TEST SYSTEM NUMBER AND MULTIPLY VULNERABLE FLAG
C
         IF ((IY(J) .LT. 0) .OR. (IY(J) .GT. 10)) IE(11) = ICHK
```

```
          IF ((IU(J). LT. 0) .OR. (IU(J) .GT. 1)) IE(12) = ICHK
          IF ((IY(J) .EQ. 0) .AND. (IU(J) .NE. 0)) THEN
            IE(11) = ICHK
            IE(12) = ICHK
          END IF
C
CF    TEST DEPTH OF PENETRATION AND ASSOCIATE PROBABILITIES OF KILL
C
          IF ((NOPNTS(J) .LT. 2) .AND. (NOPNTS(J) .GT. 10)) IE(13) = ICHK
          IF (DEPTH(1,J) .LT. 0.0) IE(14) = ICHK
          IF ((PKVAL(1,J) .LT. 0.0) .OR. (PKVAL(1,J) .GT. 1.0))
     $      IE(15) = ICHK
          IX = 2
  165     CONTINUE
            IF ((DEPTH(IX,J) .LT. 0) .OR. (DEPTH(IX,J) .LT. DEPTH(IX-1,J)))
     $        IE(12+2*IX) = ICHK
            IF ((PKVAL(1,J) .LT. 0.0) .OR. (PKVAL(1,J) .GT. 1.0) .OR.
     $        (PKVAL(IX,J) .LT. PKVAL(IX-1,J))) IE(13+2*IX) = ICHK
            IX = IX + 1
          IF ((IX .LE. 10) .AND. (IX .LE. NOPNTS(J))) GO TO 165
C
CF $ STEP D
CF    PLACE AN ASTERISK IN FRONT OF LINES WITH ERRORS
C
          DO 170 I = 1,33
            IF (IE(I) .NE. IBLNK) THEN
              IE(34) = IASTER
              GO TO 180
            END IF
  170     CONTINUE
C
CF    PRINT COMPONENT INFORMATION WITH ERRORS FLAGGED
C
  180     WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),IE(2),
     $      MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),IE(6),
     $      THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),IE(12),
     $      NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),PKVAL(K,J),
     $      IE(13+2*K),K = 1,NOPNTS(J))
C
CF    IF ALL COMPONENTS TESTED? THEN * ELSE 120
C
  199 CONTINUE
C
CF $ STEP E
CF    PRINT CRITICAL COMPONENT SUMMARY
C
        WRITE (IWR,200)
        WRITE (IWR,101)
C
        DO 210 I = 1,34
          IE(I) = IBLNK
  210 CONTINUE
C
      DO 235 I = 1,3
        NTYPE = 0
        DO 220 J = 1,NOCOMP
          IF (IFG(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $        IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $        IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $        IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $        PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
          END IF
  220   CONTINUE
C
        WRITE (IWR,230) NTYPE,I
  235 CONTINUE
C
CF    PRINT COMPONENT LISTING BY MATERIAL TYPE
C
      WRITE (IWR,300)
      WRITE (IWR,101)
C
      DO 330 I = 1,11
        NTYPE = 0
        DO 310 J = 1,NOCOMP
          IF (MAT(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $        IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $        IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $        IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $        PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
          END IF
  310   CONTINUE
C
        WRITE (IWR,320) NTYPE,I
  330 CONTINUE
C
CF    PRINT COMPONENT LISTING BY TABLE LOOK-UP INDEX
C
      WRITE (IWR,400)
      WRITE (IWR,101)
C
      DO 430 I = 1,9
        NTYPE = 0
        DO 410 J = 1,NOCOMP
          IF (ITAB(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $        IE(2),MAT(J),IE(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $          IE(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IF(11),IU(J),
     $          IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $          PKVAL(K,J),IF(13+2*K),K = 1,NOPNTS(J))
            END IF
  410   CONTINUE
C
        WRITE (IWR,420) NTYPE,I
  430 CONTINUE
C
CF    PRINT COMPONENT LISTING BY ANALYSIS TYPE
C
      WRITE (IWR,500)
      WRITE (IWR,101)
C
      DO 530 I = 0,2
        NTYPE = 0
        DO 510 J = 1,NOCOMP
          IF (IANA(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $          IE(2),MAT(J),IF(3),ITAB(J),IF(4),IANA(J),IF(5),TINIT(J),
     $          IF(6),THINFL(J),IE(9),RHOF(J),IE(10),IY(J),IE(11),IU(J),
     $          IE(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),IE(12+2*K),
     $          PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
            END IF
  510   CONTINUE
C
        WRITE (IWR,520) NTYPE,I
  530 CONTINUE
C
CF    PRINT COMPONENT LISTING BY SYSTEM NUMBER
C
      WRITE (IWR,600)
      WRITE (IWR,101)
C
      DO 630 I = 0,10
        NTYPE = 0
        DO 610 J = 1,NOCOMP
          IF (IY(J) .EQ. I) THEN
            NTYPE = NTYPE + 1
            WRITE (IWR,190,IOSTAT=IOS) IE(13),ICOMP(J),IE(1),IFG(J),
     $          IF(2),MAT(J),IF(3),ITAB(J),IE(4),IANA(J),IE(5),TINIT(J),
     $          IE(6),THINFL(J),IE(9),PHOF(J),IE(10),IY(J),IE(11),IU(J),
     $          TF(12),NOPNTS(J),IE(13),J,(DEPTH(K,J),TE(12+2*K),
     $          PKVAL(K,J),IE(13+2*K),K = 1,NOPNTS(J))
            END IF
  610   CONTINUE
C
        WRITE (IWR,620) NTYPE,I
  630 CONTINUE
C
```

```
CF    EXIT STOP
C
      STOP
C
  100 FORMAT ('1 THE FOLLOWING DATA HAS BEEN READ AND CHECKED FOR ',
     $     'ERRORS.'/'   AN ASTERISK IN THE LEFTMOST COLUMN INDICATES A',
     $     ' POSSIBLE ERROR IN THAT LINE.'/'   THE CHARACTER ',1H<,
     $     ' IS USED TO POINT TO THE POSSIBLY ERRONEOUS ITEM.')
  101 FORMAT (//
     1          '  ICOMP    IFG       MAT        ITAB       IANA     TINIT',
     2          5X,'THINFL      RHOF        IY          IU        NOPNTS',
     3          4X,'CARD'/
     4          9X,'DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)',
     5          2X,'PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)'/
     6          9X,'DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)'
     7          2X,'PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)'/)
  190 FORMAT (1X,A1,I5,A1,I6,A1,3(I9,A1),F11.2,A1,2(F9.2,A1),
     1          I7,A1,I9,A1,I10,A1,I9,2(/F16.2,A1,9(F9.2,A1)))
  200 FORMAT ('1 CRITICAL COMPONENT SUMMARY')
  230 FORMAT (/' THERE ARE ',I4,' CRITICAL COMPONENTS HAVING CRITICALI',
     $  'TY FLAG ',I3/)
  300 FORMAT ('1 COMPONENT LISTING BY MATERIAL TYPE')
  320 FORMAT (/' THERE ARE ',I4,' COMPONENTS OF MATERIAL TYPE ',I3/)
  400 FORMAT ('1 COMPONENT LISTING BY TABLE-LOOKUP INDEX')
  420 FORMAT (/' THERE ARE ',I4,' COMPONENTS HAVING TABLE INDEX ',I3/)
  500 FORMAT ('1 COMPONENT LISTING BY ANALYSIS TYPE')
  520 FORMAT (/' THERE ARE ',I4,' COMPONENTS OF ANALYSIS TYPE ',I3/)
  600 FORMAT ('1 COMPONENT LISTING BY SYSTEM NUMBER')
  620 FORMAT (/' THERE ARE ',I4,' COMPONENTS IN SYSTEM NUMBER ',I3/)
C
CF    FINISH
C
      END
      SUBROUTINE PROPY
C
CF START PROPY
CF TITLE                    QKLOOK:  SUBROUTINE PROPY
CF ENTER PROPY
C
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $            TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $            RHOF(500),IRVRS,IU(500),IY(500)
      COMMON    THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON   /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON   /PROP/ ALP,RHO,SVAL,TMLT,XLAMBD,RATE,JCOMP,N,DPM,CVAL,
     $            TVAP,CPL,ITER,T
      DIMENSION CONDUC(11,2,10),SPECIF(11,2,10),DENSIT(11),TMELT(11,2),
     $            HEATFU(11),ALPHA(11,16),ALPHAT(11,9)
C
C*****.....MATERIALS PROPEPTY DATA.....*****
C
C ALL MATERIALS PROPFRTY DATA IS VERY INITIAL.
C BARE MATERIALS-COUPLING COEFFFICIENT IS CONSIDERFD TO BE A CONSTANT.
C COATED/PAINTED MATERTALS-COUPLING COEFFICIENT IS A FUNCTION OF INCIDENT
C FLUX AND MATERIALS THICKNESS.
C COUPLING COEFFICIENT DATA IS FOR NORMAL ANGLE OF INCIDENCE.
C ALPHA IS COUPLING COEFFICIENT AS A FUNCTION OF INCIDENT FLUX. ALPHAT
C IS THE CORRECTION FACTOR FOR MATERIAL THICKNESS.
C
C
C 2024 AL PAINTED SURFACE
C
      DATA CONDUC(1,1,1)/6./
      DATA (CONDUC(1,1,J),J=2,10)/76.,166.,279.,339.,429.,458.,3*0./
      DATA (CONDUC(1,2,J),J=2,10)/132.,156.,185.,189.,175.,169.,3*0./
      DATA SPECIF(1,1,1)/7./
      DATA (SPECIF(1,1,J),J=2,10)/ 20.,93.,205.,316.,400.,427.,450.,2*0./
      DATA (SPECIF(1,2,J),J=2,10)/849.,907.,966.,1025.,1096.,1129.,1154.,
     $       2*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
      DATA ALPHA(1,1)/.86/
      DATA (ALPHA(1,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $       .244,.224,.158,.136,.115,.1024,.094/
      DATA ALPHAT(1,1)/1.13/
      DATA (ALPHAT(1,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
      DATA DENSIT(1)/2770./
      DATA (TMELT(1,J),J=1,2)/502.,2500./
      DATA HEATFU(1)/389112./
C
C 7075 AL PAINTED SURFACE
C
      DATA CONDUC(2,1,1)/7./
      DATA (CONDUC(2,1,J),J=2,10)/3.,107.,207.,260.,336.,401.,428.,2*0./
      DATA (CONDUC(2,2,J),J=2,10)/122.,141.,177.,179.,177.,170.,167.,
     $       2*0./
      DATA SPECIF(2,1,1)/6./
      DATA (SPECIF(2,1,J),J=2,10)/0.,100.,200.,300.,400.,450.,3*0./
      DATA (SPECIF(2,2,J),J=2,10)/820.,903.,966.,1038.,1129.,1197.,3*0./
      DATA DENSIT(2)/2800./
      DATA HEATFU(2)/380744./
      DATA (TMELT(2,J),J=1,2)/476.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 2024 AL
C
      DATA ALPHA(2,1)/.86/
      DATA (ALPHA(2,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $       .244,.224,.158,.136,.115,.1024,.094/
      DATA ALPHAT(2,1)/1.13/
      DATA (ALPHAT(2,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C 5456 (AMG6) AL PAINTED SURFACE
C
      DATA CONDUC(3,1,1)/7./
      DATA (CONDUC(3,1,J),J=2,10)/0.,50.,100.,150.,200.,250.,300.,2*0./
      DATA (CONDUC(3,2,J),J=2,10)/109.,120.,128.,131.,132.,135.,136.,
     $       2*0./
      DATA SPECIF(3,1,1)/3./
      DATA (SPECIF(3,1,J),J=2,10)/100.,300.,650.,6*0./
      DATA (SPECIF(3,2,J),J=2,10)/920.,1046.,1255.,6*0./
      DATA DENSIT(3)/3000./
      DATA HEATFU(3)/389112./
      DATA (TMELT(3,J),J=1,2) /571.,1000./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 2024 AL
C
      DATA ALPHA(3,1)/.86/
      DATA (ALPHA(3,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $       .244,.224,.158,.136,.115,.1024,.094/
      DATA ALPHAT(3,1)/1.13/
      DATA (ALPHAT(3,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C 6AL4V TI PAINTED SURFACE
C
      DATA CONDUC(4,1,1)/7./
      DATA (CONDUC(4,1,J),J=2,10)/149.,260.,371.,427.,480.,538.,649.,
     $        2*0. /
      DATA (CONDUC(4,2,J),J=2,10)/9.5,11.8,14.,15.1,16.3,17.3,19.6,2*0./
      DATA SPECIF(4,1,1)/6./
      DATA (SPECIF(4,1,J),J=2,10)/94.,205.,316.,427.,538.,649.,3*0./
      DATA (SPECIF(4,2,J),J=2,10)/564.,594.,619.,640.,661.,678.,3*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
```

```
      DATA ALPHA(4,1)/.86/
      DATA (ALPHA(4,J),J=2,16)/.66,.51,.42,.35,.32,.29,.28,.26,.25,.24,
     $      .23,.23,.23,.23,.23/
      DATA ALPHAT(4,1)/1.59/
      DATA (ALPHAT(4,J),J=2,9)/1.04,.98,.98,.96,.96,.91,.87,.83/
      DATA (TMELT(4,J),J=1,2)/1565.,2277./
      DATA DENSIT(4)/4430./
      DATA HEATFU(4)/426870./
C
C PURE TI PAINTED SURFACE
C
      DATA CONDUC(5,1,1)/9./
      DATA (CONDUC(5,1,J),J=2,10)/ 27.,127.,227.,427.,627.,827.,1227.,
     $     1527.,1627./
      DATA (CONDUC(5,2,J),J=2,10)/21.9,20.4,19.7,19.4,20.2,21.3,24.5,
     $     27.1,28.0/
      DATA SPECIF(5,1,1)/7./
      DATA (SPECIF(5,1,J),J=2,10)/27.,127.,327.,627.,827.,927.,1127.,
     $      2*0./
      DATA (SPECIF(5,2,J),J=2,10)/422.,570.,677.,728.,743.,695.,695.,
     $     2*0./
      DATA (TMELT(5,J),J=1,2)/1660.,0./
      DATA HEATFU(5)/435136./
      DATA DENSIT(5)/4500./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 6AL4V TI
C
      DATA ALPHA(5,1)/.86/
      DATA (ALPHA(5,J),J=2,16)/.66,.51,.42,.35,.32,.29,.28,.26,.25,.24,
     $      .23,.23,.23,.23,.23/
      DATA ALPHAT (5,1)/1.59/
      DATA (ALPHAT(5,J),J=2,9)/1.04,.98,.98,.96,.96,.91,.87,.83/
C
C VM65-1 (ZK60) MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(6,1,1)/1./
      DATA (CONDUC(6,1,J),J=2,10)/25.,8*0./
      DATA (CONDUC(6,2,J),J=2,10)/109.,8*0./
      DATA SPECIF(6,1,1)/8./
      DATA (SPECIF(6,1,J),J=2,10)/152.,227.,327.,427.,520.,521.,627.,
     $     727.,0./
      DATA (SPECIF(6,2,J),J=2,10)/1063.,1075.,1205.,1031.,1380.,1305.,
     $     1364.,1599.,0./
      DATA HEATFU(6)/326352./
      DATA DENSIT(6)/1830./
      DATA (TMELT(6,J),J=1,2)/520.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(6,1)/.86/
      DATA (ALPHA(6,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $        .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(6,1)/1.13/
      DATA (ALPHAT(6,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C MLS (AZ81A,AZ80) MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(7,1,1)/3./
      DATA (CONDUC(7,1,J),J=2,10)/0.,100.,203.,6*0./
      DATA (CONDUC(7,2,J),J=2,10)/64.9,73.6,80.8,6*0./
      DATA DENSIT(7)/1800./
      DATA HEATFU(7)/338904./
      DATA (TMELT(7,J),J=1,2)/490.,0./
      DATA SPECIF(7,1,1)/7./
      DATA (SPECIF(7,1,J),J=2,10)/127.,227.,327.,427.,507.,627.,727.,
     $        2*0./
      DATA (SPECIF(7,2,J),J=2,10)/1054.,1121.,1167.,1205.,1222.,1426.,
     $        1426.,2*0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(7,1)/.86/
      DATA (ALPHA(7,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $        .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(7,1)/1.13/
      DATA (ALPHAT(7,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C A731B MAG ALLOY PAINTED SURFACE
C
      DATA CONDUC(8,1,1)/3./
      DATA (CONDUC(8,1,J),J=2,10)/127.,234.,332.,6*0./
      DATA (CONDUC(8,2,J),J=2,10)/94.,101.,107.,6*0./
      DATA (TMELT(8,J),J=1,2)/605.,1107./
      DATA DENSIT(8)/1770./
      DATA HEATFU(8)/338904./
      DATA SPECIF(8,1,1)/8./
      DATA (SPECIF(8,1,J),J=2,10)/152.,227.,327.,527.,555.,556.,627.,
     $        727.,0./
      DATA (SPECIF(8,2,J),J=2,10)/1107.,1167.,1247.,1440.,1443.,1373.,
     $        1401.,1448.,0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SIMILAR TO PAINTED 2024 AL
C
      DATA ALPHA(8,1)/.86/
      DATA (ALPHA(8,J),J=2,16)/.72,.60,.46,.40,.35,.32,.29,.27,
     $        .244,.224,.158,.136,.118,.1072,.10/
      DATA ALPHAT(8,1)/1.13/
      DATA (ALPHAT(8,J),J=2,9)/1.07,.91,.84,.75,.68,.54,.38,.03/
C
C EI435 (IMONIC 75) NICKEL-CHROMIUM ALLOY PAINTED SURFACE
C
      DATA CONDUC(9,1,1)/7./
      DATA (CONDUC(9,1,J),J=2,10)/100.,200.,400.,600.,800.,1027.,1392.,
     $        2*0./
      DATA (CONDUC(9,2,J),J=2,10)/13.9,15.7,19.1,22.6,26.0,29.3,35.8,
     $        2*0./
      DATA HEATFU(9)/322168./
      DATA DENSIT(9)/8400./
      DATA (TMELT(9,J),J=1,2)/1400.,0./
      DATA SPECIF(9,1,1)/6./
      DATA (SPECIF(9,1,J),J=2,10)/100.,300.,500.,600.,750.,800.,3*0./
      DATA (SPECIF(9,2,J),J=2,10)/468.,502.,512.,623.,606.,619.,3*0./
C
C NO DATA ALPHA AND ALPHAT ASSUMED SAME AS PAINTED 304 STAINLESS STEEL
C
      DATA ALPHA(9,1)/.86/
      DATA (ALPHA(9,J),J=2,16)/.59,.43,.34,.30,.26,.22,.19,.18,.17,.16,
     $        .12,.12,.12,.12,.12/
      DATA ALPHAT(9,1)/1.91/
      DATA (ALPHAT(9,J),J=2,9)/1.04,.98,.96,.96,.96,.96,.96,.96/
C
C 304 (1KH18N9T) STAINLESS STEEL PAINTED SURFACE
C
      DATA DENSIT(10)/7990./
      DATA HEATFU(10)/297064./
      DATA (TMELT(10,J),J=1,2)/1400.,0./
      DATA CONDUC(10,1,1)/7./
      DATA (CONDUC(10,1,J),J=2,10)/200.,300.,400.,500.,650.,1027.,1392.,
     $        2*0./
      DATA (CONDUC(10,2,J),J=2,10)/18.,19.5,20.,21.3,25.,29.5,34.7,2*0./
      DATA SPECIF(10,1,1)/5./
      DATA (SPECIF(10,1,J),J=2,10)/200.,400.,600.,800.,1093.,4*0./
      DATA (SPECIF(10,2,J),J=2,10)/519.,553.,573.,598.,669.,4*0./
C
C ALPHA AND ALPHAT REPRESENT CURVE FITS TO INITIAL AFWL DATA
C
      DATA ALPHA(10,1)/.86/
      DATA (ALPHA(10,J),J=2,16)/.59,.43,.34,.30,.26,.22,.19,.18,.17,.16,
     $        .12,.12,.12,.12,.12/
      DATA ALPHAT(10,1)/1.91/
      DATA (ALPHAT(10,J),J=2,9)/1.04,.98,.96,.96,.96,.96,.96,.96/
C
C CU BARE SURFACE
C
      DATA CONDUC(11,1,1)/5./
      DATA (CONDUC(11,1,J),J=2,10)/89.,232.,411.,449.,499.,4*0./
      DATA (CONDUC(11,2,J),J=2,10)/403.,394.,388.,387.,386.,4*0./
      DATA SPECIF(11,1,1)/9./
      DATA (SPECIF(11,1,J),J=2,10)/93.,204.,316.,427.,538.,760.,871.,
     $        982.,1065./
      DATA(SPECIF(11,2,J),J=2,10)/397.,402.,414.,423.,440.,473.,493.,
     $        513.,540./
      DATA DENSIT(11)/8960./
      DATA (TMELT(11,J),J=1,2)/1083.,2595./
      DATA HEATFU(11)/211710./
C
C  ALPHA IS A GUESS
C
      DATA (ALPHA(11,J),J=1,16)/.4,15*0./
C
C
      MATL = MAT(N)
C
C           IF ITER .NE.0, FIRST ITERATION OF TEMPERATURE HAS BEEN DONE,
C           ONLY NEED TO RE-EVALUATE THOSE PROPERTIES WHICH
C           ARE TEMPERATURE DEPENDENT
C
CF $ STEP A
CF   IF ENCOUNTER PROPERTIES ALREADY COMPUTED? THEN 900
C
      IF (ITER .EQ. 0) THEN
C
CF   GET VAPOR TEMPERATURE, MELTING TEMPERATURE, AND HEAT OF FUSION
CF * FOR THE MATERIAL TYPE
C
        T = TINIT(N)
        TVAP = TMELT(MATL,2)
        TMLT = TMELT(MATL,1)
        XLAMBD = HEATFU(MATL)
C
C           COATED OR BARE MATERIALS - GO TO 250 FOR COATED MATERIALS.
C
CF   IF MATERIAL IS BARE COPPER? THEN * ELSE 250
C
        IF (MATL .GE. 11) THEN
C
CF   GET COUPLING COEFFICIENT FOR COPPER
C
          ALP = ALPHA(MATL,1)
        ELSE
C
CF   INTERPOLATE THE INTENSITY DEPENDENT FACTOR  ALPF
C
          IF (PEAKF .GE. 1.0E+04) THEN
            XVAL = 1.0E+04
            K = 11
  260       CONTINUE
              K = K + 1
              AVAL = K - 10
            IF (PEAKF .GT. XVAL*AVAL) GO TO 260
          ELSE
            XVAL = 1.0E+03
            K = 1
  310       CONTINUE
```

```
              K = K + 1
              AVAL = K - 1
            IF (PEAKF .GE. XVAL*AVAL) GO TO 310
          END IF
          YVAL = ALPHA(MATL,K-1) - ALPHA(MATL,K)
          EM = -YVAL / XVAL
          ALPF = ALPHA(MATL,K) + EM * (PEAKF - XVAL * (AVAL))
C
C           THICKNESS CORRECTION METHODOLOGY IS AN APPROXIMATION SCHEME
C           REPRESENTING ONLY A FIRST LEVEL APPROACH.
C
CF    INTERPOLATE THE THICKNESS DEPENDENT FACTOR  ALPT
C
          DP = DPM / 0.0254
          IF (DP .GE. 2.0E-01) THEN
            ALPT = ALPHAT(MATL,9)
          ELSE
            IF (DP .GE. 5.0E-02) THEN
              XVAL = 5.0E-02
              K = 6
  360         CONTINUE
                K = K + 1
                AVAL = K - 5
              IF (DP .GE. XVAL*AVAL) GO TO 360
            ELSE
              XVAL = 1.0E-02
              K = 1
  410         CONTINUE
                K = K + 1
                AVAL = K - 1
              IF (DP .GE. XVAL*AVAL) GO TO 410
            END IF
            YVAL = ALPHAT(MATL,K-1) - ALPHAT(MATL,K)
            EM = -YVAL/XVAL
            ALPT = ALPHAT(MATL,K) + EM * (DP - XVAL * (AVAL))
          END IF
C
CF    COMPUTE THE COUPLING COEFFICIENT AS THE PRODUCT OF ALPF AND ALPT
C
          ALPC = ALPF * ALPT
          IF ((ALPC .LT. ALPHA(MATL,1)) .AND. (ALPC .GT.
     $      ALPHA(MATL,16))) THEN
            ALP = ALPC
          ELSE IF (ALPC .GE. ALPHA(MATL,1)) THEN
            ALP = ALPHA(MATL,1)
          ELSE
            ALP = ALPHA(MATL,16)
          END IF
C
CF    GET THE COMPONENT DENSITY FOR THE MATERIAL TYPE
C
```

```
            END IF
            RHO = DENSIT(MATL)
            TEMP = (TMLT + T) / 2.
            T = TEMP
C
CF $ STEP B
CF   INTERPOLATE THE TEMPERATURE DEPENDENT PROPERTIES:
C
           END IF
           TEMP = T
C
CF * THERMAL CONDUCTIVITY AND
C
           MVAL = NINT(CONDUC(MATL,1,1)) + 1
           IF (TEMP .LE. CONDUC(MATL,1,2)) THEN
             CVAL = CONDUC(MATL,2,2)
           ELSE IF (TEMP .GE. CONDUC(MATL,1,MVAL)) THEN
             CVAL = CONDUC(MATL,2,MVAL)
           ELSE
             DO 50 I = 2,MVAL
               IF (TEMP .LE. CONDUC(MATL,1,I)) THEN
                 CVAL = CONDUC(MATL,2,I) - (CONDUC(MATL,1,I) - TEMP) *
     $             (CONDUC(MATL,2,I) - CONDUC(MATL,2,I-1)) /
     $             (CONDUC(MATL,1,I) - CONDUC(MATL,1,I-1))
                 GO TO 100
               END IF
   50        CONTINUE
           END IF
C
CF * SPECIFIC HEAT
C
  100      MVAL = NINT(SPECIF(MATL,1,1)) + 1
           IF (TEMP .LE. SPECIF(MATL,1,2)) THEN
             SVAL = SPECIF(MATL,2,2)
           ELSE IF (TEMP .GE. SPECIF(MATL,1,MVAL)) THEN
             SVAL = SPECIF(MATL,2,MVAL)
           ELSE
             DO 150 I = 2,MVAL
               IF (TEMP .LE. SPECIF(MATL,1,I)) THEN
                 SVAL = SPECIF(MATL,2,I) - (SPECIF(MATL,1,I) - TEMP) *
     $             (SPECIF(MATL,2,I) - SPECIF(MATL,2,I-1)) /
     $             (SPECIF(MATL,1,I) - SPECIF(MATL,1,I-1))
                 GO TO 200
               END IF
  150        CONTINUE
           END IF
  200      CONTINUE
C
           RETURN
C
CF   EXIT RETURN
CF    FINISH
C
      END
```

```
C              PROGRAM QKPK
C
CF START QKPK
CF TITLE                        QKLOOK:  PROGRAM QKPK
CF ENTER QKPK
C
C        THIS VERSION ACCEPTS LOS OUTPUT DIRECTLY FROM FALCON RESEARCH AND
C        DEVELOPMENT CO. PARALLEL RAY SHOTLINE GENERATING PROGRAM FASTGEN
C        DATED OCT. 1973.
C
         INTEGER    PKNBR
         COMMON     ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
        $             TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
        $             RHOF(500),IRVRS,TU(500),IY(500)
         COMMON     THINFL(500),SH(2,170),JH(5,170),CINCH
         COMMON    /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
         COMMON    /TWO/ XLOS(100),JHT(100),NENC,OBQ(100)
         COMMON    /THREE/ PKTIME(10,200),PLS(200),I1(200)
         COMMON    /FOUR/ RATES(100,25),FTIM(25),FXCM(25),FXM(25),IFMAX,
        $             TIMAX(25),MAXT(27),NTMAX
         COMMON    /LUNITS/ IRD,IWR,IIN,IOUT
         COMMON    /SIZES/ ITC,IFX,IFXX
         DIMENSION THRAT1(500),THRAT2(500)
         DATA       ITC,MAXENC,PKNBR,IFX,IFXX /500,100,10,25,27/
         DATA       IRD,IWR,IIN,IOUT /5,6,1,2/
         DATA       MAXT /27*0/
         DATA       MAT(499)/1/,MAT(500)/2/
         DATA       IFG(499)/0/,IFG(500)/0/
         DATA       ITAB(499)/0/,ITAB(500)/0/
         DATA       IANA(499)/2/,IANA(500)/1/
         DATA       TINIT(499)/110./,TINIT(500)/35./
         DATA       IY(499)/0/,IY(500)/0/
         DATA       RHOF(499),RHOF(500)/1.,1./
         DATA       IECHO /1/
C
         OPEN (IRD,FILE='QKPKDATA',RECFM='DS',MAXRECL=80,PAD='YES')
         OPEN (IWR,FILE='QKPKPRINT',RECFM='DS',CARRIAGE CONTROL='FORTRAN')
         OPEN (IIN,FILE='QKPKTAPEIN',FORM='UNFORMATTED',RECFM='VARIABLE')
         OPEN (IOUT,FILE='QKPKTAPEOUT',FORM='UNFORMATTED',RECFM='VARIABLE')
C
C              TO SUPPRESS ECHO OF QKPK OUTPUT FILE, SET IECHO = 0
C
C              ITC  IS THE DIMENSION OF THE COMPONENT ARRAYS
C              IFX  IS THE DIMENSION OF THE FLUX ARRAYS
C              IFXX  IS IFX + 2
C
C             DATA STATEMENTS FOR LOCATIONS ITC-1 AND ITC (LAST FOR ICOMP,
C             MAT, IANA, ETC.) ARE FOR AL 2024 AND AL 7075, RESPECTIVELY.
C             ANY 0XXX COMPONENT NUMBER THAT CAN BE REPRESENTED BY THE DATA
C             IN LOCATION ITC-1 NEED NOT BE INPUT.  LIKEWISE 7XXX COMPONENT
C             MAY USE DEFAULT VALUES IN LOCATION ITC.
```

```
C
CF $ STEP A
CF   EXECUTE RDATA  READS THE QKLOOK FORMATTED DATA DECK
C
       CALL RDATA
C
CF   IF DECK INCLUDED TIME INTERVALS? THEN **1
C
       IF (NTMAX .LE. 0) THEN
C
C            IF NO TIMES FOR SHOTLINE COUNTS WERE READ BY RDATA,
C            USE THE TIMES FROM THE FLUX DISTRIBUTION
C
         NTMAX = IFMAX
C
CF   ASSIGN FLUX DISTRIBUTION TIMES AS DEFAULT TIME INTERVALS FOR
CF * SHOT LINE BREAKDOWN
C
         DO 3 I = 1,NTMAX
           TIMAX(I) = FTIM(I)
    3    CONTINUE
C
       END IF
C
C           IY IS THE SYSTEM NUMBER.  E.G., 1 COULD BE USED FOR ALL
C           MULTIPLY-VULNERABLE COMPONENTS IN THE PROPULSION SYSTEM, 2
C           COULD BE USED FOR FLIGHT CONTROLS, ETC.
C           VULNERABLE AREAS WILL BE DEVELOPED AND STORED IN ARRAY
C           PMULT WITHIN SUBROUTINE PENT.
C
CF $ STEP B
CF   READ VIEWING PLANE DESCRIPTION
C
       READ (TIN) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,RADEXP
C
C             IF CINCH IS .LE. 0., ASSUME INPUT ALREADY IN INCHES
C
       IF (CINCH .LE. 0.) CINCH = 1.0
       CMETER = CINCH*0.0254
C
C            CONVERT GRID SPACING TO INCHES
C
       GRID = GRID * CINCH
       YMAX = YMAX * CINCH
       YMIN = YMIN * CINCH
       ZMAX = ZMAX * CINCH
       ZMIN = ZMIN * CINCH
       IF (IRVRS .EQ. 1) THEN
         Y1  = -YMAX
         YMAX = -YMIN
         YMIN = Y1
```

```
            EL = -EL
            AZ = AZ - 180.
            IF (AZ .LT. 0.) AZ = AZ + 360.
          END IF
C
CF    EXECUTE FSORT  SORTS THE COMPONENT INFORMATION ARRAYS INTO
CF *  ASCENDING ORDER BY COMPONENT NUMBER
C
          CALL FSORT(ICOMP,NOCOMP,MAT,IFG,ITAB,IANA,TINIT,NOPNTS,DEPTH,
         $  PKVAL,ITC,PKNBR,THINFL,RHOF,IY,IU)
C
C                THESE ARRAYS ARE NOW SORTED IN ASCENDING ORDER OF COMPONENT
C                NUMBERS -- PUT ON FILE IOUT
C
CF    PRINT THE COMPONENT INFORMATION ARRAYS
C
          WRITE (IWR,1003)
C
          DO 50 J = 1,NOCOMP
            WRITE (IWR,1004) ICOMP(J),IFG(J),MAT(J),ITAB(J),IANA(J),
         $    TINIT(J),THINFL(J),RHOF(J),IY(J),IU(J),J,(DEPTH(I,J),
         $    PKVAL(I,J),I=1,NOPNTS(J))
C
C                  CONVERT INPUT LENGTHS TO METERS
C
            DO 40 I = 1,NOPNTS(J)
              DEPTH(I,J) = DEPTH(I,J) * CMETER
   40       CONTINUE
            THINFL(J) = THINFL(J) * CMETER
   50 CONTINUE
C
          NENC = 0
          IFILL = 0
          IFM = IFMAX - 1
          IFXM = IFXX - 1
C
CF    WRITE VIEWING PLANE DESCRIPTION, COMPONENT DATA, AND FLUX TABLE
C
          WRITE (IOUT) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
          WRITE (IOUT) (ICOMP(I),IFG(I),IY(I),IU(I),NOPNTS(I),(PKVAL(J,I),
         $  J=1,NOPNTS(I)),I=1,NOCOMP),IPVRS,IFMAX,(FTIM(I),FXCM(I),I=1,
         $  IFMAX)
          IF (IECHO .EQ. 1) THEN
              WRITE (IWR,1000) IOUT
              WRITE (IWR,1014) AZ,EL,GRID,IDVEH,YMAX,YMIN,ZMAX,ZMIN,NOCOMP
              WRITE (IWR,1015)
              DO  60 I = 1,NOCOMP
                WRITE (IWR,1016) ICOMP(I),IFG(I),IY(I),IU(I),
         $        NOPNTS(I),(PKVAL(J,I),J=1,NOPNTS(I))
   60         CONTINUE
              WRITE (IWR,1017) IPVRS,IFMAX,(FTIM(I),FXCM(I),I = 1,IFMAX)
```

```
      END IF
C
CF $ STEP C
CF   READ SHOT LINE DATA ON NEXT RECORD
C
  100 CONTINUE
      READ (IIN) (DUM,(SH(I,J),I=1,2),(JH(I,J),I=1,5),J=1,170)
C
CF $ STEP D
CF   LOOP TO PROCESS EACH SET OF LOS DATA
C
  101 CONTINUE
        DO 110 J = 1,170
C
C           JH(2,J) = 0 SIGNALS END OF VIEW
C
CF   IF END OF VIEW? THEN 5000
C
          IF (JH(2,J) .EQ. 0) GO TO 5000
          ICODE = MOD(JH(1,J),10)
          NENC = NENC + 1
C
C          DIMENSIONS ARE SET FOR A MAX OF 100 COMPONENTS PER SHOT LINE
C          ATTEMPTS TO LOAD ABOVE THIS RESULTS IN A STOP
C
CF   IF MORE THAN 100 ENCOUNTERS ON THE SHOT LINE? THEN 530
C
          IF (NENC .GT. 100) GO TO 530
C
C          LOS IN METERS
C
          XLOS(NENC) = FLOAT(JH(3,J)) * 0.01 * CMETER
C
C          COMPONENT NAME
C
          JHT(NENC) = JH(2,J)
C
C          OBLIQUITY
C
          IF (IRVRS .EQ. 0) THEN
            OBQ(NENC) = FLOAT(JH(4,J)) * 0.001
          ELSE
            OBQ(NENC) = FLOAT(JH(5,J)) * 0.001
          END IF
C
C           ICODE = 9 SIGNALS END OF SHOT LINE
C
CF   IF END OF SHOT LINE? THEN * ELSE 110
C
          IF (ICODE .EQ. 9) THEN
C
```

```
CF $ STEP E
CF   IF WANT SHOT LINE DIRECTION REVERSED? THEN * ELSE **1
CF   EXECUTE REVRSE  REVERSES THE ORDER OF THE COMPONENTS ON THE
CF * SHOT LINE
C
           IF (IRVRS .EQ. 1) CALL REVRSE(J)
C
CF   EXECUTE RAT  COMPUTES PENETRATION RATES FOR EACH ENCOUNTER
CF * AT EACH FLUX LEVEL
C
           DO 170 IR = 1,IFMAX
             PEAKF = FXCM(IR)
             FLX = FXM(IR)
             CALL RAT (RATES(1,IR),IR,MAXENC)
             IF (IFLAG .EQ. 0) GO TO 180
  170      CONTINUE
C
C          THE VALUES OF NENC (NO. OF ENCOUNTERS), NCRIT (NO. OF
C          CRITICAL ENCOUNTERS), AND IFLAG (=0 IF NO CRITICALS)
C          ARE SET BY SUBROUTINE RAT.
C
C          SUBROUTINE RAT ALSO CHANGES JHT TO THE LOCATION OF
C          THE COMPONENT IN ARRAY ICOMP, AND XLOS TO THE ADJUSTED
C          THICKNESS IN METERS (ADJUSTED BY RHOF OR THINFL)
C
C          PKTIME AND I1 FIRST CONTAIN  Y  Z  NCRIT
C          (Y AND Z POSITIONS ARE NOT CONVERTED TO INCHES )
C                         A ZERO NCRIT INDICATES NO CRITICAL COMPONENT
C                         ON THE SHOT LINE
C
CF $ STEP F
CF   STORE THE SHOT LINE COORDINATES AND NUMBER OF ENCOUNTERS IN THE
CF * BINARY OUTPUT ARRAYS
C
  180      IFILL = IFILL + 1
           PKTIME(1,IFILL) = SH(1,J) * CINCH
           PKTIME(2,IFILL) = SH(2,J) * CINCH
           DO 190 K = 3,PKNBR
             PKTIME(K,IFILL) = 0.0
  190      CONTINUE
           PLS(IFILL) = 0.
           I1(IFILL) = NCRIT
C
C          MAXT(IFXX) COUNTS THE NUMBER OF NON-CRITICAL SHOT LINES
C
           IF (NCRIT .EQ. 0) MAXT(IFXX) = MAXT(IFXX) + 1
C
CF   IF BINARY OUTPUT ARRAYS FULL? THEN * ELSE **1
C
           IF (IFILL .EQ. 200) THEN
C
```

```
CF    WRITE BINARY OUTPUT ARRAYS
C
                    WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),I1(JX),
     $                JX=1,200)
                    IF (IECHO .EQ. 1)
     $                WRITE (IWR,1002) ((PKTIME(IX,JX),IX=1,10),PLS(JX),
     $                I1(JX),JX=1,200)
                    IFILL = 0
                  END IF
C
CF    IF CRITICAL ENCOUNTERS ON SHOT LINE? THEN * ELSE 110
C
                  IF (IFLAG .NE. 0) THEN
C
                    TUSED = 0.
                    TMAX = 0.
                    TMAX1 = 1.E30
                    IRUSED = 1
C
CF $ STEP G
CF    LOOP FOR EVERY ENCOUNTER ON THE SHOT LINE
C
  401               CONTINUE
                    DO 499 II = 1,NENC
C
CF    COMPUTE AND STORE TIME NEEDED TO PENETRATE TO DEPTH XLOS
C
                      RBAR = 0.
                      IRNOW = IRUSED
                      TNOW = TUSED
                      DISLFT = XLOS(II)
  410                 CONTINUE
                        RATE = RATES(II,IRNOW)
                        IF (RATE .LE. 0.) GO TO 430
                        IF (DISLFT .LE. (FTIM(IRNOW) - TNOW) * RATE) GO TO 420
                        DISLFT = DISLFT - (FTIM(IRNOW) - TNOW)*RATE
                        TNOW = FTIM(IRNOW)
                        IRNOW = IRNOW + 1
                      IF (IRNOW .LE. IFMAX) GO TO 410
                      IRNOW = IRNOW - 1
  420                 CONTINUE
                      TNOW = TNOW + DISLFT / RATE
                      TLOS = TNOW - TUSED
                      RBAR = XLOS(II) / TLOS
  430                 CONTINUE
                      I = JHT(II)
                      IF (IFG(I) .NE. 0) THEN
                        IFILL = IFILL + 1
                        PLS(IFILL) = TNOW
                        I1(IFILL) = JHT(II)
                        THRAT1(I) = AMAX1(THRAT1(I),DEPTH(1,I) * OBO(II) /
     $                  XLOS(II))
                      THRAT2(I) = AMAX1(THRAT2(I),DEPTH(NOPNTS(I),I) *
     $                  OBQ(II) / XLOS(II))
C
C               THE NEXT NENC ENTRIES (ONE FOR EACH CRITICAL COMPONENT) ARE
C                 PKTIME(1)    TIME NEEDED TO PENETRATE TO PKMN FROM
C                              START OF SHOT LINE
C                 PKTIME(I)    TIME NEEDED TO PENETRATE TO A AN INTERMEDIATE
C                              PK BETWEEN PKMN AND PKMX (I IN [2,N-1])
C                 PKTIME(N)    TIME NEEDED TO PENETRATE TO PKMX FROM
C                              START OF SHOT LINE (N IN [2,10])
C                 PLS          TIME NEEDED TO PENETRATE TO XLOS FROM
C                              START OF SHOT LINE
C                 I1           THE LOCATION (IE SUBSCRIPT) OF THIS COMP ID
C                              IN ICOMP
C
CF    COMPUTE AND STORE TIME NEEDED TO PENETRATE TO DEPTH PKMN
C
                      DO 440 K = 1,10
                        PKTIME(K,IFILL) = 0.0
  440                 CONTINUE
                      IF (RBAR .GT. 0.) THEN
                        DO 450 K = 1,NOPNTS(I)
                          PKTIME(K,IFILL) = DEPTH(K,I) * OBQ(II) / RBAR +
     $                      TUSED
  450                   CONTINUE
C
C               TMAX1 IS THE EARLIEST TIME THAT THE MAXIMUM SHOT LINE
C               PK WILL OCCUR
C
                        IF ((PKTIME(NOPNTS(I),IFILL) .LE. PLS(IFILL))
     $                    .AND. (PKVAL(NOPNTS(I),I) .EQ. 1.0))
     $                    TMAX1 = AMIN1(TMAX1,PKTIME(NOPNTS(I),IFILL))
C
C               TMAX IS THE MINIMUM TIME NEEDED TO PROCESS ALL
C               COMPONENTS ON THIS SHOT LINE.
C
                        IF (PKVAL(NOPNTS(I),I) .EQ. 1.0) THEN
                          TMAX = AMAX1(TMAX,AMIN1(TNOW,PKTIME(NOPNTS(I),
     $                      IFILL)))
                        ELSE
                          TMAX = AMAX1(TMAX,AMIN1(TNOW,PLS(IFILL)))
                        END IF
C
CF    IF BINARY OUTPUT ARRAYS FULL? THEN * ELSE *+1
C
                        IF (IFILL .EQ. 200) THEN
C
CF    WRITE BINARY OUTPUT ARRAYS
C
                          WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),
     $                   I1(JX),JX=1,200)
                       IF (IFCHO .EQ. 1)
     $                   WRITE (IWR,1002) ((PKTIME(IX,JX),IX=1,10),
     $                     PLS(JX),I1(JX),JX=1,200)
                       IFILL=0
                     END IF
                   END IF
                 END IF
                 TUSED = TNOW
                 IRUSED = IRNOW
C
CF    IF LAST ENCOUNTER ON SHOT LINE? THEN * ELSE 401
C
  499          CONTINUE
C
C             FIND APPROPRIATE TIME BIN FOR TMAX1 AND INCREMENT
C             THAT BIN.  IF TMAX1 .GT. LAST BIN TIME, INCREMENT MAXT(IFXM)
C
CF $ STEP H
CF    FIND THE EARLIEST TIME INTERVAL FOR MAXIMUM SHOT LINE PK
C
               TMAX1 = AMIN1(TMAX,TMAX1)
               DO 500 I=1,NTMAX
                 IF (TMAX1 .LE. TIMAX(I)) THEN
                   MAXT(I) = MAXT(I) + 1
                   GO TO 510
                 END IF
  500          CONTINUE
               MAXT(IFXM) = MAXT(IFXM) + 1
  510          CONTINUE
C
               NENC=0
C
CF $ STEP I
CF    IF MORE LOS DATA ON THIS RECORD? THEN 101 ELSE 100
C
             END IF
           END IF
  110    CONTINUE
C
      GO TO 100
C
C            END OF VIEW
C
CF $ STEP J
CF    WRITE LAST RECORD OF BINARY OUTPUT WITH END OF VIEW FLAG
C
 5000 CONTINUE
      IFILL = IFILL + 1
C
      DO 120 J = IFILL,200
```

```
         I1(J) = 9999
  120 CONTINUE
C
C             END OF THE VIEW WILL BE SENSED BY A 9999 FOR NENC, THE
C             NUMBER OF ITEMS ON THE SHOT LINE
C
        WRITE (IOUT) ((PKTIME(IX,JX),IX=1,10),PLS(JX),I1(JX),JX=1,200)
        IF (IECHO .EQ. 1) THEN
           WRITE (IWR,1002) ((PKTIME(IX,JX),IX=1,10),
     $        PLS(JX),I1(JX),JX=1,IFILL-1)
           WRITE (IWR,1001)
        END IF
C
CF    PRINT BREAKDOWN OF CRITICAL SHOT LINES BY TIME NEEDED
CF  * TO REACH MAXIMUM PK
C
        WRITE (IWR,1005) MAXT(IFXX)
        NSL = MAXT(IFXM)
        TE = 0.
        JF = 1
        FE = FXCM(1)
C
        DO 520 I = 1,NTMAX
          TB = TE
          TE = TIMAX(I)
          FB = FE
C
          JFF = JF
          DO 518 J = JFF,IFM
            JF = J
            IF (TE .LE. FTIM(J)) GO TO 519
            IF (TE .LE. FTIM(J+1)) THEN
              JF = J + 1
              GO TO 519
            END IF
  518     CONTINUE
          JF = IFMAX
  519     FE = FXCM(JF)
          WRITE (IWR,1006) TB,TE,FB,FE,MAXT(I)
          NSL = NSL + MAXT(I)
  520 CONTINUE
        TB = TE
        TE = 1.E30
        FB = FE
        FE = FXCM(IFMAX)
        WRITE (IWR,1006) TB,TE,FB,FE,MAXT(IFXM)
        WRITE (IWR,1011) NSL
        NSL = NSL + MAXT(IFXX)
        WRITE (IWR,1012) NSL
C
CF    PRINT LIST OF COMPONENTS WITH AN LOS THICKNESS LESS THAN
CF * PKMN OR PKMX
C
      WRITE (IWR,1009)
C
      DO 600 J = 1,NOCOMP
        IF (THRAT1(J) .GT. 1.0) THEN
          WRITE (IWR,1010) ICOMP(J),THRAT1(J),THRAT2(J)
        ELSE IF (THRAT2(J) .GT. 1.0) THEN
          WRITE (IWR,1013) ICOMP(J),THRAT2(J)
        END IF
  600 CONTINUE
C
CF   EXIT STOP
C
      STOP
C
CF $ STEP K
CF   PRINT FATAL ERROR MESSAGE, TOO MANY ENCOUNTERS
C
  530 CONTINUE
      WRITE (IWR,1008) NENC
C
CF   EXIT STOP
C
      STOP
C
 1000 FORMAT ('1 THE FOLLOWING IS AN ECHO OF THE DATA WRITTEN ON THE ',
     $  'QKPK OUTPUT FILE ( LOGICAL UNIT ',I2,' )'/)
 1001 FORMAT ('0END OF ECHO OF DATA WRITTEN ON QKPK OUTPUT FILE')
 1002 FORMAT ('0PKTIME(J,I), PLS(I), II(I), J=1,10, I=1,200'/
     $  200(11F10.5,I8/))
 1003 FORMAT ('1VALUES USED FOR THIS RUN'//
     $         ' ICOMP    IFG        MAT         ITAB        IANA      TINIT',
     $         5X,'THINEL      RHOF        IY          IU          CARD'/
     $         8X,'DEPTH(1)  PKVAL(1)  DEPTH(2)  PKVAL(2)  DEPTH(3)',
     $         2X,'PKVAL(3)  DEPTH(4)  PKVAL(4)  DEPTH(5)   PKVAL(5)'/
     $         8X,'DEPTH(6)  PKVAL(6)  DEPTH(7)  PKVAL(7)  DEPTH(8)'
     $         2X,'PKVAL(8)  DEPTH(9)  PKVAL(9)  DEPTH(10)  PKVAL(10)'/)
 1004 FORMAT (I6,I7,3I10,F11.2,2F10.2,I9,I10,I11/2(F15.2,9F10.2/))
 1005 FORMAT ('1 TOTAL NUMBER OF NON-CRITICAL SHOTLINES =',I7//
     $  '0BREAKDOWN OF CRITICAL SHOTLINES BY TIME NEEDED TO REACH ',
     $  'MAXIMUM PK'/'0BEGIN TIME',5X,'END TIME',5X,'BEGIN FLUX',5X,
     $  'END FLUX',5X,'NUMBER OF SHOTLINES')
 1006 FORMAT (1X,F10.2,3X,F10.2,5X,F10.2,3X,F10.2,10X,I10)
 1008 FORMAT ('0NENC =',I10,' WHICH IS GREATER THAN 100'/
     $  '0...PROGRAM HALTING')
 1009 FORMAT ('1 THE FOLLOWING COMPONENTS HAVE AT LEAST ONE SHOTLINE',
     $  ' ENCOUNTER WHERE THE THICKNESS IS LESS THAN PKMN OR PKMX....'/
     $  6X,'IF THE THICKNESS IS LESS THAN PKMN, COMPONENT PK IS ZERO ',
     $  'FOR THAT ENCOUNTER'/6X,'IF THE THICKNESS IS LESS THAN PKMX, '
     $  'COMPONENT PK IS LESS THAN 1.0 FOR THAT ENCOUNTER'/'0COMPONENT',
     $  5X,'MAXIMUM VALUE OF PKMN/THICKNESS',5X,'MAXIMUM VALUE OF ',
     $   'PKMX/THICKNESS')
 1010 FORMAT (1X,I6,20X,F10.3,25X,F10.3)
 1011 FORMAT ('0',25X,'TOTAL NUMBER OF CRITICAL SHOTLINES =',I10)
 1012 FORMAT (/'0TOTAL NUMBER OF SHOTLINES =',I10)
 1013 FORMAT (1X,I6,21X,'LESS THAN 1.0',21X,F10.3)
 1014 FORMAT ('0AZ, EL, GRID, IDVEH, YMAX, YMIN, ZMAX, ZMIN, NOCOMP'/
     $  3F10.5,I8,4F10.5,I8)
 1015 FORMAT ('0ICOMP(I), IFG(I), IY(I), IU(I), NOPNTS(I), (PKVAL(J,I)',
     $  'J=1,NOPNTS(I)),I=1,NOCOMP')
 1016 FORMAT (5I6,10F8.2)
 1017 FORMAT ('0IRVRS, IFMAX, (FTIM(I), FXCM(I), I=1,IFMAX',
     $  2I10/4(2F10.3,5X))
C
CF    FINISH
C
      END
      SUBROUTINE RAT (RATES,IR,MAXENC)
C
      DIMENSION LOC(100),RATES(MAXENC)
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $            TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $            RHOF(500),IRVRS,IU(500),IY(500)
      COMMON    THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON   /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON   /TWO/ XLOS(100),JHT(100),NENC,OBQ(100)
      COMMON   /PROP/ ALP,RHX,CP,TMLT,XLAMBD,RATE,JCOMP,J,DP,XK,TVAP,
     $            CPL,ITER,T
      COMMON   /SIZES/ ITC,IFX,IFXX
C
C           THIS ROUTINE COMPUTES THE PENETRATION RATES FOR EACH
C           COMPONENT ALONG A SHOT LINE.
C
CF $ STEP A
CF    IF FIRST RATE COMPUTATION FOR THIS SHOT LINE? THEN * ELSE 110
      IF (IR .EQ. 1) THEN
        IFLAG = 0
        NCRIT = 0
        DO 100 I = 1,NENC
          JCOMP = JHT(I)
          CALL BSRCH(ICOMP,NOCOMP,ITC)
          LOC(I) = J
          IF (IFG(J) .NE. 0) THEN
C
CF     COUNT THE CRITICAL COMPONENTS
C
            NCRIT = NCRIT + 1
            IFLAG = I
          END IF
  100   CONTINUE
C
        NENC = IFLAG
C
C           IFLAG POINTS TO THE LAST CRITICAL COMPONENT ON THE SHOTLINE
C           IF IFLAG = 0, THERE ARE NO CIRITICAL COMPONENTS ON
C           THIS SHOT LINE.
C
CF    IF CRITICAL COMPONENTS ON SHOT LINE? THEN 110 ELSE *
CF    EXIT RETURN
C
        IF (IFLAG .EQ. 0) RETURN
      END IF
CF $ STEP B
CF    LOOP FOR EVERY ENCOUNTER ON THE SHOT LINE
C
  110 CONTINUE
C
C           LOOP THROUGH THE COMPONENTS
```

```
C
         DO 300 I = 1,NENC
            J = LOC(I)
            DP = XLOS(I)
C
CF    IF FIRST RATE COMPUTATION FOR THIS SHOT LINE? THEN * ELSE 131
C
            IF (IR .EQ. 1) THEN
C
C                 THINFL IS THE NORMAL THICKNESS REPLACEMENT
C                 (USUALLY TWICE THE WALL THICKNESS FOR TUBES MODELLED
C                 IN THE INFLUENCE MODE)
C
CF    COMPUTE ENCOUNTERED COMPONENT THICKNESS
C
               IF (THINFL(J) .GT. 0.) THEN
C
C                USE A CONSTANT NORMAL THICKNESS FOR THIS COMPONENT INSTEAD OF
C                A RATIO
C
                  DP = THINFL(J) * OBQ(I)
               ELSE
                  DP = XLOS(I) * RHOF(J)
               END IF
               XLOS(I) = DP
            END IF
            RATE = 0.
C
C              COMPONENTS HAVING ZERO THICKNESS ARE ASSIGNED A ZERO
C              PENETRATION RATE.  (THIS MAKES THE COMPONENT 'TRANSPARENT'
C              ...IT NEED NOT BE PENETRATED AND CANNOT BE DAMAGED)
C
            IF (DP .GT. 0.) THEN
C
CF * STEP C
CF    IF USE TABLE LOOK-UP CODE? THEN 400
C
  131 CONTINUE
               IF (ITAB(J) .EQ. 0) THEN
                  ITER = 0
C
CF    EXECUTE PROPY  GET THE COMPONENT PROPERTIES
C
                  CALL PROPY
                  CPS = CP
                  RHO = RHX
C
C                 IAMA IS THE ANALYSIS TYPE
C                    = 1 FOR MELT IN PLACE
C                    = 2 FOR MELT REMOVED
C
```

```
CF   IF MELT IN PLACE ANALYSIS? THEN * ELSE 500
C
            IF (IANA(J) .EQ. 1) THEN
              ITER = 1
              T = TMLT
C
CF   COMPUTE SURFACE TEMPERATURE
C
  335         CONTINUE
                CALL PROPY
                TSTAR = TMLT + FLX * ALP * DP / XK
              IF (ABS(T-TSTAR) .GE. 100.) THEN
                T = TSTAR
                GO TO 335
              END IF
C
CF   COMPUTE PENETRATION RATE USING MELT IN PLACE ANALYSIS
C
              IF ((TSTAR .GT. TVAP) .AND. (TVAP .GT. 0.)) THEN
                STAR = XK * (TVAP - TMLT) / DP
                RATE = STAR / (RHO * (CPS * (TMLT - TINIT(J)) +
     $            XLAMBD + CP * .5 * (TVAP - TMLT)))
              ELSE
                RATE = FLX * ALP / (RHO * (CPS * (TMLT - TINIT(J)) +
     $            XLAMBD + CP * (.5 * (TSTAR + TMLT) - TMLT)))
              END IF
            ELSE
C
CF   COMPUTE PENTERATION RATE USING MELT REMOVED ANALYSIS
C
              RATE = FLX * ALP / (RHO * (CP * (TMLT - TINIT(J)) +
     $          XLAMBD))
            END IF
          ELSE
C
CF   EXECUTE TABLE  GET PENETRATION RATE BY TABLE LOOK-UP
C
            CALL TABLE
          END IF
        END IF
C
CF * STEP D
CF   STORE PENETRATION RATE IN RATES ARRAY
C
        RATES(I) = RATE
        JHT(I) = J
C
CF   IF LAST SHOT LINE ENCOUNTER? THEN * ELSE 110
C
  300 CONTINUE
C
```

```
CF    EXIT RETURN
CF    FINISH
C
      RETURN
      END
      SUBROUTINE RDATA
C
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $        TINIT(500),NOPNTS(500),DFPTH(10,500),PKVAL(10,500),
     $        RHOF(500),IRVRS,TU(500),IY(500)
      COMMON  THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON /TWO/ XLOS(100),JHT(100),NENC,OBO(100)
      COMMON /FOUR/ RATES(100,25),FTIM(25),FXCM(25),FXM(25),IFMAX,
     $        TIMAX(25),MAXT(27),NTMAX
      COMMON /LUNITS/ IRD,IWR,IIN,IOUT
      COMMON /SIZES/ ITC,IFX,IFXX
C
C           THIS ROUTINE READS THE COMPONENT MATERIALS DATA CARDS
C
      READ (IRD,1000) NOCOMP
C
C           ITC MUST BE AT LEAST TWO GREATER THAN NOCOMP
C           TO ALLOW FOR THE DEFAULTS TO 2024 AL AND 7075 AL
C
      IF (ITC .GE. NOCOMP+2) THEN
C
C           NOCOMP  IS NUMBER OF COMPONENTS
C           IFMAX   IS NO. OF POINTS IN FLUX DISTRIBUTION
C           FXCM       FLUX IN WATTS/CM2
C           FXM     IS FLUX IN WATTS/M2
C           FTIM    IS TIME THROUGH WHICH THE CORRESPONDING FLUX APPLIES
C           IRVRS   IS THE SWITCH FOR REVERSING THE SHOTLINE DIRECTION
C           CINCH   IS THE FACTOR FOR CONVERSION FROM INPUT
C                      UNITS TO INCHES
C
        READ (IRD,1000) IRVRS,CINCH
        READ (IRD,1000) IFMAX
        READ (IRD,1002) (FXCM(I),I=1,IFMAX)
        READ (IRD,1002) (FTIM(I),I=1,IFMAX)
        WRITE (IWR,2000) NOCOMP,IRVRS,CINCH
        WRITE (IWR,2001) IFMAX
        TE = 0.
C
        DO 110 I = 1,IFMAX
          TB = TE
          TE = FTIM(I)
          WRITE (IWR,2002) TB,TE,FXCM(I)
  110   CONTINUE
C
C           LAST FLUX ALSO APPLIES FOR ALL TIMES BEYOND THE LAST
C           TIME READ IN.
C
        TB = TE
        TE = 1.E30
        WRITE (IWR,2002) TB,TE,FXCM(IFMAX)
```

```
C
C             CHECK IF FLUX IS WITHIN BOUNDS
C
         DO 160 I = 1,IFMAX
           IF ((FXCM(I) .GT. 0.) .AND. (FXCM(I) .LE. 6.0E+04)) THEN
C
C             CONVERT FLUX TO WATTS/M2
C
             FXM(I) = FXCM(I) * 10000.
           ELSE
             WRITE (IWR,100) FXCM(I)
             STOP
           END IF
   160   CONTINUE
C
         DO 40 I = 1,NOCOMP
C
C             IY = SYSTEM NO. FOR EACH COMPONENT IN THAT PARTICULAR SYSTEM
C             IU = 0, IF IT IS A SINGLY VULNERABLE COMPONENT
C             IU = 1, IF IT IS A MULTPLY VULNERABLE COMPONENT(NOTE: THIS
C             WILL NOT ALLOW THIS COMP. VA TO CONTRIBUTE TO THE TOTAL VA,
C             NOR TO THE SYSTEM VA)
C
C             THINFL IS THE NORMAL THICKNESS REPLACEMENT
C             (USUALLY TWICE THE WALL THICKNESS FOR TUBES
C             MODELLED IN INFLUENCE MODE)
C
           READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $       IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
           IF (IOS .EQ. 0) THEN
             IF ((NOPNTS(I) .LT. 2) .OR. (NOPNTS(I) .GT. 10)) THEN
               WRITE (IWR,1010) I,NOPNTS(I)
               STOP
             ELSE IF (NOPNTS(I) .GT. 4) THEN
               READ (IRD,1001,IOSTAT=IOS)
             END IF
           END IF
           IF (IOS .GT. 0) THEN
             WRITE (IWR,1015) I
             WRITE (IWR,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $         IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
             STOP
           ELSE IF (IOS .LT. 0) THEN
             WRITE (IWR,1020,IOSTAT=IOS) I-1,NOCOMP
             STOP
           END IF
    40   CONTINUE
C
         REWIND (IRD)
         DO 45 I = 1,5
           READ(IRD,1001)
   45   CONTINUE
C
        DO 50 I = 1,NOCOMP
        READ (IRD,1001) ICOMP(I),MAT(I),IFG(I),ITAB(I),IANA(I),
     $    TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I),(DEPTH(J,I),
     $    PKVAL(J,I), J = 1,NOPNTS(I))
   50   CONTINUE
C
C             READ TIME BINS FOR SHOTLINE MAX-TIME COUNTS.
C             (IF NOT INCLUDED IN INPUT, FLUX TIME POINTS WILL BE USED.)
C
        READ (IRD,1000,END=60) NTMAX
        READ (IRD,1002) (TTMAX(I),I=1,NTMAX)
   60   CONTINUE
      ELSE
        WRITE (IWR,2004) NOCOMP
        STOP
      END IF
C
      RETURN
C
  100 FORMAT(/5X,'***** INPUT ERROR ***** FLUX =',E10.2/
     $         'PROGRAM HALTING')
 1000 FORMAT (I5,2E10.2)
 1001 FORMAT (I5,I3,1X,I1,I2,13X,I1,F5.0,32X,F3.2,7X,F3.2,2I2/
     $         I5,9F7.4/11F7.4)
 1002 FORMAT (10F7.0)
 1010 FORMAT ('1',5X,'***** INPUT ERROR *****     AN INVALID NUMBER ',
     $         'FOR THE VARIABLE NOPNTS(I) WAS READ:'
     $         /28X,'I = ',I4,' NOPNTS(I) = ',I4)
 1015 FORMAT (/5X,'***** INPUT ERROR *****     ERROR OCCURRED DURING ',
     $         'READ OF COMPONENT CARD #',I4,' IN THE SEQUENCE'/
     $          28X,'THE FOLLOWING VALUES WERE READ:')
 1020 FORMAT (/5X,'***** INPUT ERROR *****     END OF FILE OCCURRED '
     $         'AFTER ',I4,' CARDS WERE READ,'
     $         /28X,'ALTHOUGH NOCOMP = ',I4)
 2000 FORMAT (' NUMBER OF COMPONENTS =',I5,5X,
     $         'IRVRS=',I5,5X,'FACTOR FOR CONVERSION TO INCHES=',F10.4)
 2001 FORMAT (' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5/
     $         '      BEGIN TIME',6X,'END TIME',4X,'FLUX')
 2002 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
 2004 FORMAT ('NOCOMP =',I10,' IS TOO LARGE FOR ARRAY DIMENSIONS'/
     $  '...PROGRAM HALTING')
      END
      SUBROUTINE READIN
C
      COMMON    ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $          TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $          RHOF(500),IU(500),IY(500)
      COMMON    THINFL(500),NOCOMP
      COMMON   /LUNITS/ IRD,IWR
      COMMON   /SIZES/ ITC,IFX
      DIMENSION FXCM(25),FTIM(25),TIMAX(25)
C
C           THIS ROUTINE READS THE COMPONENT MATERIALS DATA CARDS
C
      READ (IRD,1000) NOCOMP
C
      IF (ITC .GE. NOCOMP+2) THEN
C
C           NOCOMP  IS NUMBER OF COMPONENTS
C           IFMAX   IS NUMBER OF POINTS IN FLUX DISTRIBUTION
C           FXCM    IS FLUX IN WATTS/CM2
C           FTIM    IS TIME THROUGH WHICH THE CORRESPONDING FLUX APPLIES
C           IRVRS   IS THE SWITCH FOR REVERSING THE SHOTLINE DIRECTION
C           CINCH   IS THE FACTOR FOR CONVERSION FROM INPUT
C                      UNITS TO INCHES
C
        READ (IRD,1000) IRVRS,CINCH
        READ (IRD,1001) IFMAX
        WRITE (IWR,2000) NOCOMP,IRVRS,CINCH
        IF ((IFMAX .LT. 1) .OR. (IFMAX .GT. IFX))
     $    WRITE (IWR,101) IFMAX
        WRITE (IWR,2001) IFMAX
        IF (IFMAX .GT. IFX) IFMAX = IFX
        READ (IRD,1002) (FXCM(I),I=1,IFMAX)
        READ (IRD,1002) (FTIM(I),I=1,IFMAX)
        TE = 0.
C
        DO 110 I = 1,IFMAX
          TB = TE
          TE = FTIM(I)
          WRITE (IWR,2002) TB,TE,FXCM(I)
  110   CONTINUE
C
C           LAST FLUX ALSO APPLIES FOR ALL TIMES BEYOND
C           THE LAST TIME READ IN
C
        TB = TE
        TE = 1.E30
        WRITE (IWR,2002) TB,TE,FXCM(IFMAX)
C
C           CHECK IF FLUX IS WITHIN BOUNDS
C
        DO 160 I = 1,IFMAX
        IF ((FXCM(I) .LE. 0.) .OR. (FXCM(I) .GT. 6.0E+04))
     $     WRITE (IWR,100) FXCM(I)
  160   CONTINUE
C
        DO 40 I = 1,NOCOMP
C
C           IY = SYSTEM NO. FOR EACH COMPONENT IN THAT PARTICULAR SYSTEM
C           IU = 0, IF IT IS A SINGLY VULNERABLE COMPONENT
C           IU = 1, IF IT IS A MULTPLY VULNERABLE COMPONENT(NOTE: THIS
C           WILL NOT ALLOW THIS COMP. VA TO CONTRIBUTE TO THE TOTAL VA,
C           NOR TO THE SYSTEM VA)
C
C           THINFL IS THE NORMAL THICKNESS REPLACEMENT
C           (USUALLY TWICE THE WALL THICKNESS FOR TUBES
C           MODELLED IN INFLUENCE MODE)
C
          READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $     IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
          IF (IOS .EQ. 0) THEN
            IF ((NOPNTS(I) .LE. 0) .OR. (NOPNTS(I) .GT. 10)) THEN
              WRITE (IWR,1010) I,NOPNTS(I)
              STOP
            ELSE IF (NOPNTS(I) .GT. 4) THEN
              READ (IRD,1001,IOSTAT=IOS)
            END IF
          END IF
          IF (IOS .GT. 0) THEN
            WRITE (IWR,1015) I
            WRITE (IWR,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $        IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I)
            STOP
          ELSE IF (IOS .LT. 0) THEN
            WRITE (IWR,1020,IOSTAT=IOS) I-1,NOCOMP
            NOCOMP = I - 1
          END IF
   40   CONTINUE
C
        REWIND (IRD)
        DO 45 I = 1,5
          READ(IRD,1001)
   45   CONTINUE
C
        DO 50 I = 1,NOCOMP
        READ (IRD,1001,IOSTAT=IOS) ICOMP(I),MAT(I),IFG(I),ITAB(I),
     $   IANA(I),TINIT(I),THINFL(I),RHOF(I),IY(I),IU(I),NOPNTS(I),
     $   (DEPTH(J,I),PKVAL(J,I),J = 1,NOPNTS(I))
   50   CONTINUE
C
C           READ TIME BINS FOR SHOTLINE MAX-TIME COUNTS
C           (IF NOT INCLUDED IN INPUT, FLUX TIME POINTS WILL BE USED)
C
```

```
        READ (IRD,1000,END=60) NTMAX
        IF ((NTMAX .GE. 1) .AND. (NTMAX .LE. IFX)) THEN
          READ (IRD,1002) (TIMAX(I),I=1,NTMAX)
          WRITE (IWR,1005) (TIMAX(I),I=1,NTMAX)
          RETURN
        ELSE
          WRITE (IWR,102) NTMAX
          WRITE (IWR,1007)
          STOP
        END IF
      ELSE
        WRITE (IWR,1007)
        STOP
      END IF
C
   60 CONTINUE
      WRITE (IWR,1006)
      RETURN
C
  100 FORMAT(/5X,"***** INPUT ERROR ***** FLUX =",E10.2)
  101 FORMAT(/5X,"***** INPUT ERROR ***** IFMAX =",I10)
  102 FORMAT(/5X,"***** INPUT ERROR ***** NTMAX =",I10)
 1000 FORMAT(I5,2E10.2)
 1001 FORMAT(I5,I3,1X,I1,I2,13X,I1,F5.0,32X,F3.2,7X,F3.2,2I2/
     $   I5,9F7.4/11F7.4)
 1002 FORMAT (10F7.0)
 1005 FORMAT (' TIMAX VALUES ='/(10F10.2))
 1006 FORMAT (' NO TIMAX VALUES WERE READ IN ... FLUX TIME POINTS ',
     $  'WILL BE USED')
 1007 FORMAT (' ARRAY DIMENSIONS ARE TOO SMALL...PROGRAM HALTING')
 1010 FORMAT ('1',5X,'***** INPUT ERROR *****     AN INVALID NUMBER ',
     $          'FOR THE VARIABLE NOPNTS(I) WAS READ:'
     $          /28X,'I = ',I4,' NOPNTS(I) = ',I4)
 1015 FORMAT (/5X,'***** INPUT ERROR *****     ERROR OCCURRED DURING ',
     $          'READ OF COMPONENT CARD #',I4,' IN THE SEQUENCE'/
     $          28X,'THE FOLLOWING VALUES WERE READ:')
 1020 FORMAT (/5X,'***** INPUT ERROR *****     END OF FILE OCCURRED '
     $          'AFTER ',I4,' CARDS WERE READ,'
     $          /28X,'ALTHOUGH NOCOMP = ',I4)
 2000 FORMAT(' NUMBER OF COMPONENTS =',I5,5X,
     $    ' IRVRS =',I5,5X,'FACTOR FOR CONVERSION TO INCHES =',F10.4)
 2001 FORMAT (/' NUMBER OF POINTS IN FLUX DISTRIBUTION =',I5//
     $  ' BEGIN TIME',5X,'END TIME',7X,'FLUX')
 2002 FORMAT (1X,F10.2,3X,F10.2,3X,F10.2)
      END
      SUBROUTINE REVRSE(J)
C
C          ROUTINE TO REVERSE THE ORDER OF COMPONENTS ALONG A SHOTLINE
C          FOR THE L V A C  PROGRAMS
C
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $          TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $          RHOF(500),IRVRS,IU(500),IY(500)
      COMMON  THTNFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /TWO/ XLOS(100),JHT(100),NENC,OBQ(100)
C
C           Y-POSITION OF SHOTLINE MUST CHANGE SIGN IN ORDER
C           TO MAINTAIN A RIGHT-HANDED COORDINATE SYSTEM
C
      SH(1,J) = -SH(1,J)
C
      DO 30 N = 1,NENC/2
        JJ = NENC - N + 1
        J1 = JHT(N)
        S1 = OBQ(N)
        S2 = XLOS(N)
        JHT(N) = JHT(JJ)
        OBQ(N) = OBQ(JJ)
        XLOS(N) = XLOS(JJ)
        JHT(JJ) = J1
        OBQ(JJ) = S1
        XLOS(JJ) = S2
   30 CONTINUE
C
      RETURN
      END
      SUBROUTINE TABLE
      COMMON  ICOMP(500),MAT(500),IFG(500),ITAB(500),IANA(500),
     $          TINIT(500),NOPNTS(500),DEPTH(10,500),PKVAL(10,500),
     $          RHOF(500),IRVRS,IU(500),IY(500)
      COMMON  THINFL(500),SH(2,170),JH(5,170),CINCH
      COMMON /ONE/ NOCOMP,FLX,IFLAG,PEAKF,NCRIT
      COMMON /PROP/ ALP,RHO,CP,TMLT,XLAMBD,RATE,JCOMP,J,DP,XK,TVAP,
     $          CPL,ITER,T
      COMMON /LUNITS/ IRD,IWR,IIN,IOUT
C
C          THIS ROUTINE RETURNS A PENETRATION RATE USING A
C          TABULAR VALUE BASED ON THE MATERIAL TYPE.
C
      IF (ITAB(J) .EQ. 1) THEN
C
C          PLEXIGLASS
C
        RATE = 4.35E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 2) THEN
C
C          TRIPLEX DIELECTRIC
C
        RATE = 2.6822E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 3) THEN
C
C          FIBRITE
C
        RATE = 2.633E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 4) THEN
C
C          RUBBER
C
        RATE = 2.6909E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 5) THEN
C
C          GLASS
C
        RATE = .35E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 6) THEN
C
C          FIBER GLASS
C
        RATE = .44E-10 * FLX * 0.9
      ELSE IF (ITAB(J) .EQ. 7) THEN
C
C          PYROCERAM
C
C          3.74E+6 FLX EQUALS 374 W/CM-2, SINCE FLX IS IN METERS
C          IF FLX IS LESS THAN 3.74E+6, RETURN A RATE OF ZERO
C
        IF (FLX .GT. 3.74E+6) THEN
```

```
            RATE = -1.4E-4 + 3.74E-11 * FLX * 0.9
          ELSE
            WRITE (IWR,2000)
            STOP
          END IF
        ELSE IF (ITAB(J) .EQ. 8) THEN
C
C              GLASS FIBER EPOXY
C
          RATE = 0.556E-10 * FLX * 0.9
        ELSE IF (ITAB(J) .EQ. 9) THEN
C
C              GRAPHITE EPOXY
C
          RATE = 0.119E-10 * FLX * 0.9
        ELSE
          RATE = 0.0
          WRITE(IWR,1000) JCOMP,ITAB(J)
        END IF
C
        RETURN
C
 1000 FORMAT(' THIS COMPONENT DOES NOT FIT THE BOUNDS FOR TABLES',2I6)
 2000 FORMAT(' THE FLUX IS INDICATED TO BE LESS THAN 374 W/CM**2'/
     $         'OCHECK SUBROUTINE TABLE...PROGRAM HALTING')
        END
```

SUBROUTINE ASPTOIND

This subroutine is used to correlate each of the standard 26 viewing angles used by programs like FASTGEN with an index used by the program ASALT. Each of the standard azimuth-elevation angle pairs has a correspondence to one of 26 ASALT indices. The subroutine is called by the main program.

The first set of statements

```
      SUBROUTINE ASPTOIND(AZ,EL,ILOOK)
      INTEGER AZIX
      INTEGER ELIX
```

is used to pass three arguments, AZ, EL, and ILOOK, and to declare AZIX and ELIX to be integers. AZ and EL are the FASTGEN azimuth and elevation angles and ILOOK is the corresponding ASALT index.

The next statements

```
      IF (ABS(EL-90.) .LE. 1.E-05) THEN
        ILOOK = 26
      ELSE IF (ABS(EL+90.) .LE. 1.E-05) THEN
        ILOOK = 1
```

test for a top or bottom view. The block IF first checks for an elevation angle of 90 degrees, implying a top view. If this angle is found, the ASALT index is set to 26. Otherwise, a check for a bottom view with an elevation angle of -90 is made. If this test is successful, the ASALT index is set to 1; otherwise, the block IF continues with the ELSE branch where checks are made for the other 24 viewing angles.

The statements

```
      ELSE
        ELIX = 0
        CHK = -45.
        DO 10 I = 1,3
          IF (ABS(EL-CHK) .GT. 1.E-05) THEN
            CHK = CHK + 45.
            ELIX = ELIX + 8
          END IF
   10   CONTINUE
```

check for a standard elevation and set an elevation index accordingly. The first two assignment statements initialize the angle to be checked and the elevation index. The DO loop checks for the remaining three valid elevation angles. If the elevation angle and the checking angle are not equal within roundoff limits, the checking angle, CHK, is incremented by 45 degrees and the elevation index, ELIX, is incremented by 8.

The following statements

```
        IF (CHK .LT. 50.) THEN
          CHK = 0.0
          IF (AZ .LT. 180.) THEN
            AZIX = 6
          ELSE
            AZIX = 2
            AZ = AZ - 180.
          END IF
```

are used in preparation for determining the azimuth angle. The first IF statement checks to determine if a standard elevation angle was found. If not, the ELSE branch of the block IF is executed. Otherwise, the THEN branch sets the azimuth checking angle to 0.0. The second IF statement determines whether the azimuth in question is less than 180 degrees. If so, the azimuth index is set to 2 and the azimuth is adjusted so as to be less than 180 degrees.

The statements

```
            DO 30 I = 1,4
              IF (ABS(AZ-CHK) .GT. 1.E-05) THEN
                CHK = CHK + 45.
                AZIX = AZIX + 1
              END IF
   30       CONTINUE
```

form a DO loop to check for a standard azimuth angle. The statements in the IF block are executed when the azimuth value does not equal the checking angle within roundoff limits. In this case, the checking angle is incremented by 45 degrees and the azimuth index is incremented.

The next statements

```
              IF (CHK .LT. 180) THEN
                ILOOK = ELIX + AZIX
              ELSE
                WRITE (6,901) AZ, EL
              END IF
```

form a block IF to take the appropriate action depending on whether a standard azimuth is present. If a standard azimuth is found, the THEN branch is executed and the ASALT index is the sum of the elevation and azimuth indices. If not, an error message is printed.

The statements

```
              ELSE
                WRITE (6,901) AZ, EL
              END IF
            END IF
```

are executed when a standard elevation is not found. The WRITE statement prints an error message. The two END IF statements close the IF block for determining whether a valid elevation is found and for ascertaining whether a top, bottom, or other view is present.

The statements

```
            RETURN
  901         FORMAT (' *** ERROR *** CANNOT CLASSIFY LOOKANGLES AZ, EL'/
     $                 15X,2F10.2)
            END
```

return control to the calling program, define the error message format, and end the subroutine.

SUBROUTINE FINDCOMP

This subroutine is used to search an array containing critical component numbers for a specific component. If the number is found, the array element index is returned. If not, a zero is returned. This subroutine is called by the Subroutine GETXYZ.

The first set of statements

```
      SUBROUTINE FINDCOMP(IPT,ICOMP,IC,NCRIT)
      DIMENSION ICOMP(100)
      IPT = 0
      I = 0
```

is used to pass four arguments, IPT, ICOMP, IC, and NCRIT. IPT is the element of the array containing the desired component number or zero if the component is not in the array. ICOMP is the array containing critical component numbers. IC is the component number being sought and NCRIT is the number of critical components in ICOMP. The DIMENSION statement declares ICOMP to be an array. The two assignment statements initialize the returned element number and a search index to zero.

The statements

```
  100 CONTINUE
        I = I + 1
        IF (ICOMP(I) .EQ. IC) IPT = I
      IF ((I .NE. IPT) .AND. (I .LT. NCRIT)) GO TO 100
      RETURN
      END
```

search the ICOMP array for the desired component number and return control to the calling subroutine when the search is completed. The first four statements form a "REPEAT UNTIL" loop. Statement 100 forms the beginning of the loop. The assignment statement increments the index of the component array. The IF statement checks if the current element in the array matches the sought after component. If so, IPT is set to the index of the current element. The conditional GO TO statement checks whether the desired component has been found and whether the array has been completely searched. If both of these conditions are false, a branch is made to examine the next element. If either condition is true, control returns to the calling program unit.

SUBROUTINE GETXYZ

This subroutine is used to compute the locations of critical components in the ASALT aircraft coordinate system. It obtains these coordinates by averaging the shot line coordinates that intersect the components from the front, side, and bottom views. This subroutine is called by the main program.

The statements

```
SUBROUTINE GETXYZ(ICOMP,COMP,NCRIT,NAME)
CHARACTER*8 NAME(100)
DIMENSION   ICOMP(100),COMP(3,100)
DIMENSION   SH(2,170),JH(5,170),AZV(3),ELV(3)
DIMENSION   X(100),Y(100),Z(100)
DIMENSION   NX(100),NY(100),NZ(100)
DATA AZV /0.0,90.0,90.0/, ELV /0.0,0.0,-90.0/
DATA X   /100*0.0/, Y /100*0.0/, Z /100*0.0/
DATA NX  /100*0/,  NY /100*0/,  NZ /100*0/
```

are used to pass four arguments, ICOMP, COMP, NCRIT, and NAME, to declare array dimensions, and to set initial values. ICOMP is the array containing critical component numbers. COMP is an array to hold the ASALT aircraft coordinates of the critical components. NCRIT is the number of critical components. NAME is an array used to hold the names of the critical components. SH contains the shotline y- and z-coordinates that intersect the components. JH holds the component numbers or a flag indicating the end of the view. AZV and ELV indicate the azimuth and elevations used to define the three views: front, side, and bottom. X, Y, and Z are the ASALT aircraft coordinates for each critical component. NX, NY, and NZ count the number of times that each component is described using x-, y-, or z-coordinates (only two of the three coordinates are used in each viewing plane).

The statements

```
DO 150 ILOS = 1,3
  READ (ILOS) AZ, EL
  IF ((AZ .EQ. AZV(ILOS)) .AND. (EL .EQ. ELV(ILOS))) THEN
```

are used to obtain line of sight data for the three viewing planes. The DO loop is used to read the data from each plane and to start the conversion of the coordinates to the ASALT aircraft system. The READ statement is used to get the azimuth and elevation for the current view plane. The IF statement determines if the current view plane is the standard plane that the program expects. If so, the THEN branch of the IF block is executed; otherwise the ELSE branch is executed.

The statements

```
20      READ (ILOS, END=150) (DUM,(SH(I,J),I=1,2),(JH(I,J),I=1,5),
  $       J=1,170)
          DO 100 J = 1,170
            IF (JH(2,J) .EQ. 0) GO TO 150
            SY = SH(1,J)
            SZ = SH(2,J)
            IC = JH(2,J)
            CALL FINDCOMP(IPT,ICOMP,IC,NCRIT)
```

are used to read the line of sight data and to set up for the conversion of coordinates. The READ statement forms the first statement in a "REPEAT

UNTIL" loop. The READ obtains a block of data at a time. The loop will continue executing until all the shot line data for the view have been processed. The DO loop is used to convert the shot line coordinates from the data block obtained by the READ statement. The IF statement determines if the end of the view has been reached. If so, a branch to Statement 150 is made to permit the processing of any other views. This is the exit from the "REPEAT UNTIL" loop. The next three assignment statements get the shotline y-coordinate, z-coordinate, and component number. The CALL statement returns the index of the ICOMP array containing the component if the component is critical or a zero if the component is noncritical.

The statements

```
                  IF (IPT .NE. 0) THEN
                    IF (ILOS .EQ. 1) THEN
                      Y(IPT) = Y(IPT) + SY
                      NY(IPT) = NY(IPT) + 1
                      Z(IPT) = Z(IPT) + SZ
                      NZ(IPT) = NZ(IPT) + 1
                    ELSE IF (ILOS .EQ. 2) THEN
                      X(IPT) = X(IPT) - SY
                      NX(IPT) = NX(IPT) + 1
                      Z(IPT) = Z(IPT) + SZ
                      NZ(IPT) = NZ(IPT) + 1
                    ELSE
                      X(IPT) = X(IPT) - SY
                      NX(IPT) = NX(IPT) + 1
                      Y(IPT) = Y(IPT) + SZ
                      NY(IPT) = NY(IPT) + 1
                    END IF
                  END IF
 100            CONTINUE
              GO TO 20
```

are used to convert the shot line coordinates for critical components. The first IF block is executed only when a critical component is encountered. This block contains another IF block whose branches are executed depending upon whether the front, side, or bottom view is being processed. In each case, the shot line coordinates are added to or subtracted from the appropriate aircraft coordinates and a count for each addition/subtraction is kept. Statement 100 following the two IF blocks ends the DO loop which processes the shot line data block. The GO TO statement forms the end of the "REPEAT UNTIL" loop.

The following statements

```
      ELSE
        WRITE (6,11) ILOS,AZ,EL
        STOP
      END IF
 150  CONTINUE
```

are the remaining statements in the viewing plane processing loop. The ELSE statement begins the branch of the block IF that is executed when a standard viewing plane is not found. The WRITE statement prints an error message and the STOP statement terminates the program. The END IF statement closes the IF block and Statement 150 concludes the view plane processing loop.

The next statements

```
      WRITE (6,301) NCRIT
      DO 300 I = 1,NCRIT
        IF (NX(I) .NE. 0) COMP(1,I) = X(I) / FLOAT(NX(I)) * 0.0254
        IF (NY(I) .NE. 0) COMP(2,I) = Y(I) / FLOAT(NY(I)) * 0.0254
        IF (NZ(I) .NE. 0) COMP(3,I) = Z(I) / FLOAT(NZ(I)) * 0.0254
      WRITE (6,302) I,NAME(I),ICOMP(I),COMP(1,I),NX(I), COMP(2,I),
   $    NY(I),COMP(3,I),NZ(I)
  300 CONTINUE
```

are used to complete the calculation of the ASALT aircraft coordinates and to print these values out. The first WRITE statement prints the page header. The DO loop is used to calculate the average ASALT aircraft coordinates for each critical component and send these values to the output device. The three IF statements find the average x-, y-, and z-coordinate in meters. The next WRITE statement prints the coordinate values. Statement 300 closes the DO loop.

The final set of statements

```
      RETURN
 11   FORMAT (' ***ERROR***  INCORRECT LOS FILE FOR VIEW',I3,
   $          '  AZ=',F8.1,'  EL=',F8.1)
301   FORMAT ('1',22X,'--',I3,' CRITICAL COMPONENT LOCATIONS --'/
   $          '0 + + + COMPONENT + + +'/
   $          '  INDEX   NAME   NUMBER',6X,'X-COORD. SAMPLE',
   $          6X,'Y-COORD. SAMPLE',6X,'Z-COORD. SAMPLE')
302   FORMAT (1X,I4,3X,A8,I6,1X,3(F12.2,I7,2X))
      END
```

returns control to the calling program and provides the formats for the messages that are output.

SUBROUTINE NAMES

This subroutine is used to read names of components from an input file. If the input file is empty, blanks are entered for the names. The subroutine is called by the main program.

The first statements

```
      SUBROUTINE NAMES(NAME,NCRIT)
      CHARACTER*8 NAME(100)
```

are used to pass two arguments, NAME and NCRIT. NAME is an array to hold the names of the critical components and NCRIT is the number of critical components. NAME is declared to be an array of character strings of length 8.

The statements

```
      READ (7,120,END=100) (NAME(I),I=1,NCRIT)
      RETURN
```

attempt to read the names of the critical components from an input file. If the read completes successfully, control is returned to the calling program. If an end of file is encountered, the execution branches to Statement 100.

The next statements

```
  100 DO 110 I = 1,NCRIT
         NAME(I) = '
  110 CONTINUE
      RETURN
  120    FORMAT (A8)
      END
```

are used when an end of file occurs. The DO loop is executed to initialize the component names to a string of blanks. Control is then returned to the calling program.

PROGRAM VAMERGE

This program is used to produce one of the two input files required by the ASALT-I Model in assessing survivability against laser threats. This program reads the vulnerable area files created by the QKLOOK program PEAKAY, averages the component Pk's, and prints them in the ASALT input format.

The first set of statements

```
      PROGRAM VAMERGE
      PARAMETER (TDELT = 0.5, IPRINT = 2, LINLIM = 60,
     $           XFP = 0.0, YFP = 0.0, XG = 0.0, YG = 0.0,
     $           ZG = 0.0, PSI = 0.0, NATN = 1, YJITTR = 1.0,
     $           ZJITTR = 1.0, SLEWAZ = 90.0, SLEWEL = 45.0,
     $           TRKTIM = 0.0, ATTEN = 1.0, RATTEN = 1.E+30,
     $           NAIMPT = 1)
      DIMENSION FTIM(25), FXCM(25), TIMES(10), ENERGY(10)
      DIMENSION FTIM2(25), FXCM2(25), TIMES2(10), GUN(3)
      DIMENSION ICOMP(100), PAREA(100), COMPAV(100,10)
      DIMENSION COMP(3,100), AP(100,26), WIDTH(100,26), PK(10,100)
      CHARACTER*8 NAME(100),BLANK
      CHARACTER*4 YORN(2)
```

is used to assign constant values to a number of parameters used in the ASALT input file, to declare array dimensions, and to declare character variables.

The statements

```
      DATA YORN   /'NO  ', 'YES '/
      DATA BLANK  /'        '/
      DATA IOUT,IRD,IWR,IIN /4,5,6,11/
      DATA AP     /2600*-1.0/, PK/1000*0.0/
      DATA ENERGY /10*0.0/
      DATA COMP   /300*0.0/
      DATA GUN    /3*0.0/
```

are DATA statements which are used to initialize the variables whose names appear in the DATA statement lists. The first three DATA statements set the array YORN which is used to indicate whether the reverse flag is set, assign a string of blank characters to the variable BLANK, and set logical unit numbers used in the FORTRAN I/O operations. The next three DATA statements initialize arrays used to hold presented areas, Pk's, energy flux, and component coordinates. The final DATA statement sets the coordinates of the weapon location in the General Coordinate System.

The statements

```
      WRITE (IOUT,103) TDELT,IPRINT,LINLIM
      WRITE (IOUT,102) GUN,XFP,YFP,XG,YG,ZG,PSI
```

produce the first two ASALT cards. The first card contains TDELT which is the time interval between iterations of ASALT's computations, and IPRINT and LINLIM which establish the line printer output from ASALT. The second card holds the coordinates for the weapon location and the coordinates for the coordinate system's reference point.

The statements

```
      READ (IRD,101) NCRIT
      READ (IIN,END=900) AZ,EL,IFMAX,(FTIM(I),FXCM(I),I=1,IFMAX),RVRS,
     $  NOCOMP,NTIME,(TIMES(I),I=1,NTIME)
      IRVRS = RVRS + 1.
      WRITE (IWR,111) (IIN-10),AZ,EL,YORN(IRVRS),NOCOMP,NCRIT,
     $  (FTIM(I),FXCM(I),I=1,IFMAX)
      WRITE (IWR,112) (TIMES(I),I=1,NTIME)
```

read information on the first vulnerable area file and echo it on the line printer. The first READ statement ascertains the number of critical components in the QKLOOK files. The next READ statement is used to obtain a viewing plane description, the flux distribution with time intervals, the reverse flag, the number of components, and the number and breakdown of time intervals used in the computation of vulnerable areas. The two WRITE statements send this information to the printer file.

The statements

```
      WRITE (IOUT,101) IFMAX,NATN
      WRITE (IOUT,102) (FXCM(I),I=1,IFMAX)
      FTIM(IFMAX) = 1.E+30
      WRITE (IOUT,102) (FTIM(I),I=1,IFMAX)
      WRITE (IOUT,102) YJITTR,ZJITTR
      WRITE (IOUT,102) SLEWAZ,SLEWEL,TRKTIM
      WRITE (IOUT,102) ATTEN
      WRITE (IOUT,102) RATTEN
      WRITE (IOUT,102)
      WRITE (IOUT,101) NCRIT,NAIMPT
```

are used to write the laser flux emission rates for ASALT Cards 3 through 11. Card 3 contains the number of elements in the laser flux emission array and the number of elements in the atmospheric attenuation factor array. Cards 4 and 5 contain, respectively, the laser flux emission rates and their corresponding times. Card 6 has the standard deviation due to jitter of the laser beam in the Y and Z direction. Card 7 holds the maximum azimuth and elevation slewing rate for the laser weapon as well as the minimum tracking time required before the laser can fire. Cards 8 and 9 contain, respectively, the laser beam attenuation factor and the beam atmospheric attenuation factor range argument. Card 10 provides the smoke corridor end points. However, no smoke corridor is used in this program, so Card 10 is left blank. Card 11 has the number of components in the target model as well as the number of aim points on the target.

The statements

```
      ENERGY(1) = 0.0
      IF (NTIME .LT. 10) THEN
         LIM = NTIME
      ELSE
         LIM = 9
      END IF
      DO 20 I = 1,LIM
        T1 = 0.0
        FLUX = 0.0
        J = 0
 15     CONTINUE
        J = J + 1
        IF (FTIM(J) .LT. TIMES(I)) THEN
          DELT = FTIM(J) - T1
          FLUX = FLUX + (FXCM(J) * DELT)
          T1 = FTIM(J)
        END IF
       IF ((J .LT. IFMAX) .AND. (FTIM(J) .LT. TIMES(I))) GO TO 15
```

```
         DELT = TIMES(I) - T1
         FLUX = FLUX + (FXCM(J) * DELT)
         ENERGY(I+1) = FLUX * 0.001
   20 CONTINUE
        WRITE (IOUT,102) (ENERGY(I),I=1,LIM+1)
```

are used to calculate the amount of energy accumulated to cause kill probabilities for each component during a specified time interval. These results are used for ASALT Card 12. At time 0, no energy has accumulated, so the first energy argument is set to 0.0. Since ASALT uses exactly 10 energy entries in a function defining Pk at increasing energy levels, and since the first energy level is always set to 0.0, LIM, the number of energy levels to calculate from PEAKAY, is limited to a maximum of 9. The DO loop finds the energy levels for the times specified in the TIMES array. For each cycle through this loop, the first three assignment statements reset the previous time argument T1, the accumulated energy FLUX, and the flux array subscript J to 0. Statement 15 through the conditional GO TO statement form a "REPEAT UNTIL" block and is used to accumulate the energy from the flux distribution table until the flux level for the time of interest is reached. The amount of energy received at this level before the desired time is reached is then added to the previously accumulated energy flux. The energy level is then set to this flux value after it has been converted from joules/cm^2 to kilojoules/cm^2. After all the energy arguments have been calculated, they are written to the output file.

The statements

```
      CALL NAMES(NAME,NCRIT)
      DO 250 IIN = 11,36
```

are used to read the names of the critical components and to initiate a DO loop to iterate through vulnerable area files for the standard 26 different aspects.

The statements

```
        IF (IIN .NE. 11) THEN
          READ(IIN,END=900) AZ,EL,IFMAX2,(FTIM2(I),FXCM2(I),I=1,IFMAX2),
     $      RVRS,NOCOMP2,NTIME2,(TIMES2(I),I=1,NTIME2)
          IRVRS = RVRS + 1.
          WRITE (IWR,111) (IIN-10),AZ,EL,YORN(IRVRS),NOCOMP2,NCRIT,
     $      (FTIM2(I),FXCM2(I),I=1,IFMAX2)
          WRITE (IWR,112) (TIMES2(I),I=1,NTIME2)
          FTIM2(IFMAX2) = 1.E+30
          IF ((IFMAX2 .NE. IFMAX ) .OR. (NOCOMP2 .NE. NOCOMP)
     $      .OR. (NTIME2 .NE. NTIME)) GO TO 950
          DO 300 I = 1,IFMAX
            IF ((FTIM2(I) .NE. FTIM(I)) .OR. (FXCM2(I) .NE. FXCM(I)))
     $        GO TO 950
  300     CONTINUE
          DO 320 I = 1,NTIME
            IF (TIMES2(I) .NE. TIMES(I)) GO TO 950
  320     CONTINUE
        END IF
```

form an IF block which is executed for every aspect angle except for the first one. This block reads in the azimuth, elevation, flux distribution table, reverse flag, number of components, and times of interest for the next aspect angle. The block then checks the number of components, the flux distribution table, and the times of interest to insure that these values are identical to those for the first aspect angle. If any difference is found, a GO TO statement branches to Statement 950 to write an error message.

The statements

```
          CALL ASPTOIND(AZ,EL,ILOOK)
          DO 200 J = 1,NCRIT
            READ (IIN,END=900) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
            IF (NAME(J) .EQ. BLANK)
     $        WRITE (NAME(J),195) ICOMP(J)
            WRITE (IWR,113) J,NAME(J),ICOMP(J),PAREA(J),(COMPAV(J,K),
     $        K = 1,NTIME)
  200     CONTINUE
```

are used to convert the look angles to an ASALT index and to read the presented and vulnerable areas for each critical component. The CALL statement obtains the ASALT index associated with each of the 26 views. The DO loop reads the presented and vulnerable areas for each critical component, assigns the component's number to the name array when the component's name is not already in the array, and writes this information out. The component name information is used in preparing Card 13 for the ASALT program.

The statements

```
          DO 240 J = 1,NCRIT
            AP(J,ILOOK) = PAREA(J) * 0.09290304
            WIDTH(J,ILOOK) = SQRT(AP(J,ILOOK))
            PK(1,J) = 0.0
            IF (PAREA(J) .GT. 1.E-06) THEN
              DO 230 I = 2,LIM
                PK(I,J) = PK(I,J) + COMPAV(J,I-1) / PAREA(J)
  230         CONTINUE
            END IF
  240     CONTINUE
  250 CONTINUE
```

are used to prepare ASALT Cards 14 and 15. The outer DO loop iterates for every critical component for each look angle. The first assignment statement converts the presented area from square feet to square meters. The next assignment statement assumes that the presented area is square and calculates the width in meters. The third assignment statement states that the probability of kill at time zero when no energy has accumulated is zero. The IF statement checks whether the presented area for the current component is zero. If not, the DO loop in the IF block sums the Pk's for each component for all views. Statement 250 closes the loop that iterates through the 26 aspect angles.

The statements

```
  400 DO 410 J = 2,LIM
          DO 405 I = 1,NCRIT
            PK(J,I) = PK(J,I) / 26.0
  405     CONTINUE
  410 CONTINUE
```

form two nested DO loops which average the Pk's for each component from all 26 views for each time of interest.

The statements

```
          CALL GETXYZ(ICOMP,COMP,NCRIT,NAME)
          WRITE (IWR,121)
```

```
      DO 450 I = 1,NCRIT
        WRITE (IOUT,104) NAME(I),(COMP(J,I),J=1,3)
        WRITE (IOUT,102) (AP(I,J),WIDTH(I,J),J=1,26)
        WRITE (IOUT,102) (PK(J,I),J=1,10)
        WRITE (IWR,122) I,NAME(I),ICOMP(I),(J,AP(I,J),WIDTH(I,J),J=1,26)
  450 CONTINUE
```

produce ASALT Cards 13, 14, and 15 for each critical component. The CALL statement determines the location in the Aircraft Coordinate System. The first WRITE statement sends a form feed to the print file. The DO loop iterates for each critical component. The next WRITE statement forms Card 13 by listing the name and coordinates for the component. The third WRITE statement outputs the presented area and width of the presented area for Card 14. The fourth WRITE statement outputs the average Pk at each specified time for the component for Card 15. The final WRITE statement is used to print the information from Cards 13 and 14.

The following statements

```
      WRITE (IWR,123) (ENERGY(I),I=1,10)
      DO 500 I = 1,NCRIT
        WRITE (IWR,124) I,NAME(I),ICOMP(I),(PK(J,I),J=1,10)
  500 CONTINUE
      WRITE (IOUT,125)
      STOP
```

print the energy flux distribution and the information from Card 15. Additionally, a reminder to the user to finish assembling the ASALT input deck is output.

The statements

```
  900 WRITE (IWR,126) IIN,J
      STOP
```

are used to terminate the program when the end of file is reached unexpectedly when reading data on a particular aspect. The WRITE statement prints an error message.

The next statements

```
  950 WRITE (IWR,127) IIN
      STOP
```

are used to terminate the program if a difference in values is detected between different views where no difference is expected. The WRITE statement prints an error message.

The last group of statements

```
  101 FORMAT (10I8)
  102 FORMAT (10E8.2)
  103 FORMAT (E8.2,2I8)
  104 FORMAT (A8,3E8.2)
  111 FORMAT ('1',5X,'VIEW NUMBER',I3,9X,'REVERSED',9X,
     $        'NUMBER OF COMPONENTS',19X,'FLUX TABLE'/
     $        2X,'AZ =',F6.1,'  EL =',F6.1,8X,A4,11X,'TOTAL',
     $        '   CRITICAL',13X,'TIME(SEC.)  FLUX(J/SQ.CM.)'/
     $        47X,I5,I9,16X,F6.2,8X,F8.1/(77X,F6.2,8X,F8.1))
  112 FORMAT ('1',16X,' PRESENTED AREAS AND TRUE COMPONENT VULNERABLE',
     $        ' AREAS (SQUARE FEET) PER TIME INCREMENT'/
     $        '  + + + COMPONENT + + +   PRESENTED',39X,'TIME INCREMENTS
     $      ' /'  INDEX   NAME   NUMBER      AREA',7X,10F9.2)
```

```
113 FORMAT (1X,I4,3X,A8,I6,F13.5,4X,10F9.4)
121 FORMAT ('1')
122 FORMAT (' + + + COMPONENT + + +'/' INDEX   NAME   NUMBER',
   $         3('   LOOK-ANG PRESENTED AREA   WIDTH   ')/
   $         1X,I3,3X,A8,I6,1X,3(5X,'INDEX    (SQ. METERS)  (METERS)')/
   $         (19X,3(I12,1X,2F12.2)))
123 FORMAT ('1',14X,'* * *   C O M P O N E N T   P K   ',
   $         'F U N C T I O N S   F O R   A S A L T   * * *'/
   $         45X,'DAMAGING ENERGY LEVELS IN KILOJOULES/SQ.CM.'/
   $         3X,'+ + + COMPONENT + + +  !',F7.2,9F8.2/
   $         3X,'INDEX   NAME   NUMBER  !',79('-'))
124 FORMAT (1X,I5,3X,A8,I6,'    !',10(F7.2,1X))
125 FORMAT (' ALL DONE - ACC THE AIM POINTS AND FAULT TREE')
126 FORMAT (' UNEXPECTED EOF -- IIN=',I3,' COMPONENT=',I3)
127 FORMAT (' VA FILE DOES NOT MATCH -- IIN=',I3)
195 FORMAT (' COMP',I4)
    END
```

is used to define all input and output formats and to end Program VAMERGE.

```
      SUBROUTINE ASPTOIND(AZ,EL,ILOOK)
C
C     CONVERT FASTGEN LOOK AZIMUTH AND LOOK ELEVATION TO THE
C     ASALT INDEX FOR THE SAME LOOK ANGLES.
C
C       FASTGEN                     ASALT
C      AZ       EL                AZ       EL         INDEX
C     180.     -90.                0.       0.          1
C     180.     -45.                0.      45.          2
C     225.     -45.               45.      45.          3
C     270.     -45.               90.      45.          4
C     315.     -45.              135.      45.          5
C       0.     -45.              180.      45.          6
C      45.     -45.              225.      45.          7
C      90.     -45.              270.      45.          8
C     135.     -45.              315.      45.          9
C     180.       0.                0.      90.         10
C     225.       0.               45.      90.         11
C     270.       0.               90.      90.         12
C     315.       0.              135.      90.         13
C       0.       0.              180.      90.         14
C      45.       0.              225.      90.         15
C      90.       0.              270.      90.         16
C     135.       0.              315.      90.         17
C     180.      45.                0.     135.         18
C     225.      45.               45.     135.         19
C     270.      45.               90.     135.         20
C     315.      45.              135.     135.         21
C       0.      45.              180.     135.         22
C      45.      45.              225.     135.         23
C      90.      45.              270.     135.         24
C     135.      45.              315.     135.         25
C     180.      90.                0.     180.         26
C
      INTEGER AZIX
      INTEGER ELIX
C
      IF (ABS(EL-90.) .LE. 1.E-05) THEN
        ILOOK = 26
      ELSE IF (ABS(EL+90.) .LE. 1.E-05) THEN
        ILOOK = 1
      ELSE
C
C     WHICH ELEVATION?
C
        ELIX = 0
        CHK = -45.
        DO 10 I = 1,3
          IF (ABS(EL-CHK) .GT. 1.E-05) THEN
            CHK = CHK + 45.
            ELIX = ELIX + 8
```

```
              END IF
   10       CONTINUE
C
C       WHICH AZIMUTH?
C
            IF (CHK .LT. 50.) THEN
              CHK = 0.0
              IF (AZ .LT. 180.) THEN
                AZIX = 6
              ELSE
                AZIX = 2
                AZ = AZ - 180.
              END IF
              DO 30 I = 1,4
                IF (ABS(AZ-CHK) .GT. 1.E-05) THEN
                  CHK = CHK + 45.
                  AZIX = AZIX + 1
                END IF
   30         CONTINUE
              IF (CHK .LT. 140) THEN
C
C       THE INDEX IS ...
C
                ILOOK = ELIX + AZIX
              ELSE
C
C       ERROR
C
                WRITE (6,901) AZ, EL
              END IF
            ELSE
              WRITE (6,901) AZ, EL
            END IF
          END IF
C
          RETURN
C
  901     FORMAT (' *** ERROR *** CANNOT CLASSIFY LOOKANGLES AZ, EL'/
     $             15X,2F10.2)
          END
```

```
      SUBROUTINE FINDCOMP(IPT,ICOMP,IC,NCRIT)
C
      DIMENSION ICOMP(100)
C
C     SEARCH FOR COMPONENT NUMBER, IC, IN ARRAY ICOMP WHICH CONTAINS
C     NCRIT COMPONENT NUMBERS.  RETURN THE ARRAY POSITION IN IPT OR
C     THE VALUE 0 IF THE COMPONENT IS NOT THERE.
C
      IPT = 0
      I = 0
  100 CONTINUE
        I = I + 1
        IF (ICOMP(I) .EQ. IC) IPT = I
      IF ((I .NE. IPT) .AND. (I .LT. NCRIT)) GO TO 100
C
      RETURN
      END
      SUBROUTINE GETXYZ(ICOMP,COMP,NCRIT,NAME)
C
      CHARACTER*8 NAME(100)
      DIMENSION   ICOMP(100),COMP(3,100)
      DIMENSION   SH(2,170),JH(5,170),AZV(3),ELV(3)
      DIMENSION   X(100),Y(100),Z(100)
      DIMENSION   NX(100),NY(100),NZ(100)
C
      DATA AZV /0.0,90.0,90.0/, ELV /0.0,0.0,-90.0/
      DATA X   /100*0.0/, Y /100*0.0/, Z /100*0.0/
      DATA NX  /100*0/,   NY /100*0/,  NZ /100*0/
C
C     READ THE FRONT, LEFT SIDE, AND BOTTOM LINE OF SIGHT SHOT LINE
C     TARGET DESCRIPTIONS AND DETERMINE THE X, Y, AND Z COORDINATES
C     OF EACH CRITICAL COMPONENT
C
      DO 150 ILOS = 1,3
        READ (ILOS) AZ, EL
        IF ((AZ .EQ. AZV(ILOS)) .AND. (EL .EQ. ELV(ILOS))) THEN
   20     READ (ILOS, END=150) (DUM,(SH(I,J),I=1,2),(JH(I,J),I=1,5),
     *       J=1,170)
             DO 100 J = 1,170
C
C     END OF VIEW?
C
                IF (JH(2,J) .EQ. 0) GO TO 150
                SY = SH(1,J)
                SZ = SH(2,J)
                IC = JH(2,J)
                CALL FINDCOMP(IPT,ICOMP,IC,NCRIT)
C
C     IPT=0 FOR NONCRITICAL COMPONENTS (NOT IN ARRAY ICOMP)
C
                IF (IPT .NE. 0) THEN
C
C     STORE SHOT LINE COORDINATES -- DEPENDENT ON THE CURRENT VIEW
C
                  IF (ILOS .EQ. 1) THEN
C
C     FRONT VIEW -- COORDINATES ARE Y AND Z AIRCRAFT COORDINATES
C
                    Y(IPT) = Y(IPT) + SY
                    NY(IPT) = NY(IPT) + 1
                    Z(IPT) = Z(IPT) + SZ
                    NZ(IPT) = NZ(IPT) + 1
                  ELSE IF (ILOS .EQ. 2) THEN
C
C     SIDE VIEW -- COORDINATES ARE -X AND Z AIRCRAFT COORDINATES
C
                    X(IPT) = X(IPT) - SY
                    NX(IPT) = NX(IPT) + 1
                  Z(IPT) = Z(IPT) + SZ
                  NZ(IPT) = NZ(IPT) + 1
                ELSE
C
C     BOTTOM VIEW -- COORDINATES ARE -X AND Y AIRCRAFT COORDINATES
C
                  X(IPT) = X(IPT) - SY
                  NX(IPT) = NX(IPT) + 1
                  Y(IPT) = Y(IPT) + SZ
                  NY(IPT) = NY(IPT) + 1
                END IF
              END IF
  100       CONTINUE
          GO TO 20
        ELSE
          WRITE (6,11) ILOS,AZ,EL
          STOP
        END IF
C
C     END OF VIEW
C
  150 CONTINUE
C
C     COMPUTE COMPONENT LOCATIONS IN THE ASALT AIRCRAFT COORDINATE
C     SYSTEM, USE THE AVERAGE OF THE SHOT LINE COORDINATES THAT
C     INTERSECTED THEM IN THESE THREE VIEWS
C     -- ALSO CONVERT COORDINATES FROM INCHES TO METERS --
C
      WRITE (6,301) NCRIT
      DO 300 I = 1,NCRIT
        IF (NX(I) .NE. 0) COMP(1,I) = X(I) / FLOAT(NX(I)) * 0.0254
        IF (NY(I) .NE. 0) COMP(2,I) = Y(I) / FLOAT(NY(I)) * 0.0254
        IF (NZ(I) .NE. 0) COMP(3,I) = Z(I) / FLOAT(NZ(I)) * 0.0254
        WRITE (6,302) I,NAME(I),ICOMP(I),COMP(1,I),NX(I), COMP(2,I),
     $    NY(I),COMP(3,I),NZ(I)
  300 CONTINUE
C
      RETURN
C
   11 FORMAT (' ***ERROR***  INCORRECT LOS FILE FOR VIEW',I3,
     $        '  AZ=',F8.1,'  EL=',F8.1)
  301 FORMAT ('1',22X,'--',I3,' CRITICAL COMPONENT LOCATIONS --'/
     $        '0 + + + COMPONENT + + +'/
     $        '  INDEX   NAME   NUMBER',6X,'X-COORD. SAMPLE',
     $        6X,'Y-COORD. SAMPLE',6X,'Z-COORD. SAMPLE')
  302 FORMAT (1X,I4,3X,A8,I6,1X,3(F12.2,I7,2X))
      END
      SUBROUTINE NAMES(NAME,NCRIT)
C
      CHARACTER*8 NAME(100)
C
C     READ THE COMPONENT NAMES FROM LOGICAL UNIT 7 IF PROVIDED,
C     OTHERWISE INITIALIZE TO BLANKS
C
      READ (7,120,END=100) (NAME(I),I=1,NCRIT)
      RETURN
C
  100 DO 110 I = 1,NCRIT
        NAME(I) = '        '
  110 CONTINUE
C
      RETURN
  120   FORMAT (A8)
      END
```

```
C$CONTROL USLINIT, LOCATION, FILE=1-36
      PROGRAM VAMERGE
C
C     PROGRAM VAMERGE IS USED TO READ THE VULNERABLE AREA FILES
C     CREATED BY QKLOOK PROGRAM PEAKAY, AVERAGE THE COMPONENT PK'S,
C     AND PRINT THEM IN THE ASALT INPUT FORMAT (LOGICAL UNIT 4) AS
C     WELL AS IN READABLE FORM ON THE LINE PRINTER (LOGICAL UNIT, 6)
C
C        I/O LOGICAL UNIT TABLE FOR PROGRAM VAMERGE
C     FORTRAN LOGICAL UNIT NUMBER             USE
C         1, 2, 3                        TARGET SHOT LINE DESCRIPTIONS
C                                        FOR THE FRONT, LEFT, AND
C                                        BOTTOM VIEWS
C
C         4                              FILE TO BE USED FOR ASALT INPUT
C
C         5                              INPUT FOR VAMERGE, NUMBER OF
C                                        CRITICAL COMPONENTS IN QKLOOK
C                                        FILES
C
C         6                              READABLE LINE PRINTER FILE
C                                        PRINTED BY EXECUTING VAMERGE
C
C         7                              COMPONENT NAME FILE (OPTIONAL).
C
C         8, 9, 10                       NOT USED
C
C         11 THROUGH 36                  26 VULNERABLE AREA FILES FROM
C                                        QKLOOK PROGRAM PEAKAY
C
      PARAMETER (TDELT = 0.5, IPRINT = 2, LINLIM = 60,
     $           XFP = 0.0, YFP = 0.0, XG = 0.0, YG = 0.0,
     $           ZG = 0.0, PSI = 0.0, NATN = 1, YJITTR = 1.0,
     $           ZJITTR = 1.0, SLEWAZ = 90.0, SLEWEL = 45.0,
     $           TRKTIM = 0.0, ATTEN = 1.0, RATTEN = 1.E+30,
     $           NAIMPT = 1)
      DIMENSION FTIM(25), FXCM(25), TIMES(10), ENERGY(10)
      DIMENSION FTIM2(25), FXCM2(25), TIMES2(10), GUN(3)
      DIMENSION ICOMP(100), PAREA(100), COMPAV(100,10)
      DIMENSION COMP(3,100), AP(100,26), WIDTH(100,26), PK(10,100)
      CHARACTER*8 NAME(100),BLANK
      CHARACTER*4 YORN(2)
      DATA YORN   /'NO  ', 'YES '/
      DATA BLANK  /'        '/
      DATA IOUT,IRD,IWR,IIN /4,5,6,11/
      DATA AP     /2600*-1.0/, PK/1000*0.0/
      DATA ENERGY /10*0.0/
      DATA COMP   /300*0.0/
      DATA GUN    /3*0.0/
C
C     ASALT CARD 1 -- TDELT, IPRINT, LINLIM
```

```
C
      WRITE (IOUT,103) IDELT,IPRINT,LINLIM
C
C     ASALT CARD 2 -- WEAPON LOCATION, AND C.S. REFERENCE
C
      WRITE (IOUT,102) GUN,XFP,YFP,XG,YG,ZG,PSI
      READ (IRD,101) NCRIT
C
C     READ THE FIRST VULNERABLE AREA FILE FROM LOGICAL UNIT IIN
C
      READ (IIN,END=900) AZ,EL,IFMAX,(FTIM(I),FXCM(I),I=1,IFMAX),RVRS,
     $  NOCOMP,NTIME,(TIMES(I),I=1,NTIME)
      IRVRS = RVRS + 1.
      WRITE (IWR,111) (IIN-10),AZ,EL,YORN(IRVRS),NOCOMP,NCRIT,
     $  (FTIM(I),FXCM(I),I=1,IFMAX)
      WRITE (IWR,112) (TIMES(I),I=1,NTIME)
C
C     WRITE LASER FLUX EMISSION RATES, ASALT CARDS 3, 4, AND 5
C
      WRITE (IOUT,101) IFMAX,NATN
      WRITE (IOUT,102) (FXCM(I),I=1,IFMAX)
      FTIM(IFMAX) = 1.E+30
      WRITE (IOUT,102) (FTIM(I),I=1,IFMAX)
C
C     ASALT CARDS 6 AND 7 -- JITTER AND TRACKING
C
      WRITE (IOUT,102) YJITTR,ZJITTR
      WRITE (IOUT,102) SLEWAZ,SLEWEL,TRKTIM
C
C     ASALT CARDS 8 AND 9 -- ATMOSPHERIC ATTENUATION
C
      WRITE (IOUT,102) ATTEN
      WRITE (IOUT,102) RATTEN
C
C     ASALT CARD 10 -- NO SMOKE CORRIDOR
C
      WRITE (IOUT,102)
C
C     ASALT CARD 11 -- NUMBER OF COMPONENTS AND AIM POINTS
C
      WRITE (IOUT,101) NCRIT,NAIMPT
C
C     ASALT CARD 12 -- ENERGY ARGUMENTS, COMPUTE USING FXCM AND
C                      FTIM ARRAYS FROM QKLOOK FILE
C
      ENERGY(1) = 0.0
      IF (NTIME .LT. 10) THEN
         LIM = NTIME
      ELSE
         LIM = 9
      END IF
      DO 20 I = 1,LIM
        T1 = 0.0
        FLUX = 0.0
        J = 0
   15   CONTINUE
          J = J + 1
          IF (FTIM(J) .LT. TIMES(I)) THEN
            DELT = FTIM(J) - T1
            FLUX = FLUX + (FXCM(J) * DELT)
            T1 = FTIM(J)
          END IF
        IF ((J .LT. IFMAX) .AND. (FTIM(J) .LT. TIMES(I))) GO TO 15
        DELT = TIMES(I) - T1
        FLUX = FLUX + (FXCM(J) * DELT)
C
C     CONVERT FROM JOULES/SQ.CM. TO KILOJOULES/SQ.CM.
C
        ENERGY(I+1) = FLUX * 0.001
   20 CONTINUE
      WRITE (IOUT,102) (ENERGY(I),I=1,LIM+1)
C
C     READ COMPONENT NAMES IF PROVIDED, AND CONVERT LOOK-ANGLES TO
C     ASALT INDEX
C
      CALL NAMES(NAME,NCRIT)
      DO 250 IIN = 11,36
        IF (IIN .NE. 11) THEN
C
C     READ NEXT VULNERABLE AREA FILE FROM LOGICAL UNIT IIN
C
          READ(IIN,END=900) AZ,EL,IFMAX2,(FTIM2(I),FXCM2(I),I=1,IFMAX2),
     $      RVRS,NOCOMP2,NTIME2,(TIMES2(I),I=1,NTIME2)
          IRVRS = RVRS + 1.
          WRITE (IWR,111) (IIN-10),AZ,EL,YORN(IRVRS),NOCOMP2,NCRIT,
     $      (FTIM2(I),FXCM2(I),I=1,IFMAX2)
          WRITE (IWR,112) (TIMES2(I),I=1,NTIME2)
C
C     TEST TO BE SURE NEW VA FILE IS COMPATIBLE
C
          FTIM2(IFMAX2) = 1.E+30
          IF ((IFMAX2 .NE. IFMAX ) .OR. (NOCOMP2 .NE. NOCOMP)
     $      .OR. (NTIME2 .NE. NTIME)) GO TO 950
          DO 300 I = 1,IFMAX
            IF ((FTIM2(I) .NE. FTIM(I)) .OR. (FXCM2(I) .NE. FXCM(I)))
     $        GO TO 950
  300     CONTINUE
          DO 320 I = 1,NTIME
            IF (TIMES2(I) .NE. TIMES(I)) GO TO 950
  320     CONTINUE
        END IF
        CALL ASPTOIND(AZ,EL,ILOOK)
```

```
C
C     READ PRESENTED AND VULNERABLE AREAS FOR EACH CRITICAL COMPONENT
C
        DO 200 J = 1,NCRIT
          READ (IIN,END=900) ICOMP(J),PAREA(J),(COMPAV(J,K),K=1,NTIME)
          IF (NAME(J) .EQ. BLANK)
     $      WRITE (NAME(J),195) ICOMP(J)
          WRITE (IWR,113) J,NAME(J),ICOMP(J),PAREA(J),(COMPAV(J,K),
     $      K = 1,NTIME)
  200   CONTINUE
C
C     USE THE PRESENTED AND VULNERABLE AREAS IN SQ. FEET
C     TO COMPUTE PRESENTED AREA AND WIDTH IN SQ. METERS
C     ASSUME SQUARE PRESENTED AREA
C
        DO 240 J = 1,NCRIT
          AP(J,ILOOK) = PAREA(J) * 0.09290304
          WIDTH(J,ILOOK) = SQRT(AP(J,ILOOK))
          PK(1,J) = 0.0
          IF (PAREA(J) .GT. 1.E-06) THEN
C
C     SUM PK'S OVER ALL VIEWS
C
            DO 230 I = 2,LIM
              PK(I,J) = PK(I,J) + COMPAV(J,I-1) / PAREA(J)
  230       CONTINUE
          END IF
  240   CONTINUE
  250 CONTINUE
C
C     AVERAGE THE PK'S FROM ALL 26 VIEWS
C
  400 DO 410 J = 2,LIM
        DO 405 I = 1,NCRIT
          PK(J,I) = PK(J,I) / 26.0
  405   CONTINUE
  410 CONTINUE
C
C     DETERMINE THE COMPONENT CENTROID LOCATIONS
C
      CALL GETXYZ(ICOMP,COMP,NCRIT,NAME)
      WRITE (IWR,121)
C
C     ASALT CARDS 13, 14, AND 15  FOR EACH CRITICAL COMPONENT
C
      DO 450 I = 1,NCRIT
        WRITE (IOUT,104) NAME(I),(COMP(J,I),J=1,3)
        WRITE (IOUT,102) (AP(I,J),WIDTH(I,J),J=1,26)
        WRITE (IOUT,102) (PK(J,I),J=1,10)
        WRITE (IWR,122) I,NAME(I),ICOMP(I),(J,AP(I,J),WIDTH(I,J),J=1,26)
  450 CONTINUE
```

```
C
C     PRINT COMPONENT PK FUNCTIONS ON THE LINE PRINTER TOO
C
      WRITE (IWR,123) (ENERGY(I),I=1,10)
      DO 500 I = 1,NCRIT
        WRITE (IWR,124) I,NAME(I),ICOMP(I),(PK(J,I),J=1,10)
  500 CONTINUE
C
C     ALL DONE -- FORMAT 125 IS A REMINDER TO FINISH THE ASALT INPUT
C
      WRITE (IOUT,125)
      STOP
C
C     FATAL ERRORS DETECTED
C
  900 WRITE (IWR,126) IIN,J
      STOP
  950 WRITE (IWR,127) IIN
      STOP
C
C     FORMATS
C
  101 FORMAT (10I8)
  102 FORMAT (10E8.2)
  103 FORMAT (E8.2,2I8)
  104 FORMAT (A8,3E8.2)
  111 FORMAT ('1',5X,'VIEW NUMBER',I3,9X,'REVERSED',9X,
     $          'NUMBER OF COMPONENTS',19X,'FLUX TABLE'/
     $          2X,'AZ =',F6.1,'  EL =',F6.1,8X,A4,11X,'TOTAL',
     $          '   CRITICAL',13X,'TIME(SEC.)  FLUX(W/SQ.CM.)'/
     $          47X,I5,I9,16X,F6.2,8X,F8.1/(77X,F6.2,8X,F8.1))
  112 FORMAT ('1',16X,' PRESENTED AREAS AND TRUE COMPONENT VULNERABLE',
     $          ' AREAS (SQUARE FEET) PER TIME INCREMENT'/
     $          '  + + + COMPONENT + + +   PRESENTED',39X,'TIME INCREMENTS'
     $          /'  INDEX    NAME    NUMBER       AREA',7X,10F9.2)
  113 FORMAT (1X,I4,3X,A8,I6,F13.5,4X,10F9.4)
  121 FORMAT ('1')
  122 FORMAT (' + + + COMPONENT + + +'/' INDEX    NAME    NUMBER',
     $          3('    LOOK-ANG PRESENTED AREA    WIDTH    ')/
     $          1X,I3,3X,A8,I6,1X,3(5X,'INDEX     (SQ. METERS)  (METERS)')/
     $          (19X,3(I12,1X,2F12.2)))
  123 FORMAT ('1',14X,'* * *   C O M P O N E N T   P K   ',
     $          'F U N C T I O N S   F O R   A S A L T   * * *'/
     $          45X,'DAMAGING ENERGY LEVELS IN KILOJOULES/SQ.CM.'/
     $          3X,'+ + + COMPONENT + + +  !',F7.2,9F8.2/
     $          3X,'INDEX    NAME    NUMBER  !',79('-'))
  124 FORMAT (1X,I5,3X,A8,I6,'    !',10(F7.2,1X))
  125 FORMAT (' ALL DONE - ADD THE AIM POINTS AND FAULT TREE')
  126 FORMAT (' UNEXPECTED EOF -- IIN=',I3,' COMPONENT=',I3)
  127 FORMAT (' VA FILE DOES NOT MATCH -- IIN=',I3)
  195 FORMAT (' COMP',I4)
      END
```